buildpy Documentation

Release 1.8.12

Blake Huber

Jun 05, 2021

README

1 Summary **1**

2 Getting Started **3**

3 Documentation **5**

4 Supported Linux Distributions **7**

5 Help **9**

6 Author & Copyright **13**

7 Disclaimer **15**

8 License **17**

9 Dependencies **27**

10 Installation **29**
10.1 Debian, Ubuntu, Linux Mint, Ubuntu Variants . . . 29
10.2 Redhat, CentOS, Fedora RPM-based Distributions . . . 31

11 Upgrading **35**
11.1 Debian, Ubuntu Variants . . . 35
11.2 Upgrading Redhat-based Distributions . . . 36

12 Uninstall **39**
12.1 Debian and Ubuntu Variants . . . 39
12.2 Redhat, CentOS, Fedora RPM Variants . . . 39

13 Use Cases **41**
13.1 Verify Operating System Dependencies . . . 41
13.2 Compile Python 3 - Basic Installation . . . 42
13.3 Compile Python 3 - Advanced Installation . . . 42
13.4 Compile Python 3 - Specific Version . . . 43
13.5 Unattended Install . . . 43
13.6 Uninstalling Python 3 Versions . . . 44

14 Frequently Asked Questions **45**
14.1 General Questions . . . 46
14.2 Required User Permissions . . . 46
14.3 Python Version Support . . . 47
14.4 Supported Linux Operating System Distributions . . . 47
14.5 Uninstall Python3 . . . 48
14.6 Logging FAQ . . . 48
14.7 Miscellaneous Questions . . . 48

15 Screenshot Index **51**

16 Operating System Detection **53**

17 Python Binary Version Status **55**

18 OS Packages Installed Status **57**

19 Download Python Source (Default) **59**

20 Download Python Source, Named Version **61**

21 Remove Build Artifacts **63**

22 Program Information **65**

23 Log Files **69**

24 buildpy Source Code **71**
24.1 Bash Module Index . . . 71
24.2 buildpy . . . 71
24.3 colors.sh . . . 126
24.4 exitcodes.sh . . . 131
24.5 version.py . . . 132
24.6 os_distro.sh . . . 132
24.7 std_functions.sh . . . 150

25 Debian Package Creation **167**
25.1 Debian Package Creation Start . . . 167
25.2 Debian Package Creation Build . . . 168
25.3 Debian Package Final Contents . . . 169

26 RPM Package Creation **171**
26.1 RPM Package Assembly . . . 172
26.2 RPM Package Build . . . 173
26.3 Docker RPM Package Build . . . 174
26.4 RPM Package Final Contents . . . 175

27 Current Release **177**
27.1 v1.8.12 | Release Notes . . . 177

28 Release History **179**

29 Release Note Index **181**
29.1 v1.8.11 | Release Notes . . . 181
29.2 v1.8.10 | Release Notes . . . 182
29.3 v1.8.9 | Release Notes . . . 182

29.4 v1.8.8 | Release Notes . . . 183
29.5 v1.8.5 | Release Notes . . . 183
29.6 v1.8.3 | Release Notes . . . 184
29.7 v1.8.1 | Release Notes . . . 184
29.8 v1.7.20 | Release Notes . . . 185
29.9 v1.7.19 | Release Notes . . . 185
29.10 v1.7.18 | Release Notes . . . 186
29.11 v1.7.17 | Release Notes . . . 186
29.12 v1.7.16 | Release Notes . . . 186
29.13 v1.7.15 | Release Notes . . . 187

30 Module Index **189**

31 Site Index **191**

32 Search Project **193**

CHAPTER 1

Summary

- buildpy is a utility for compiling and installing on Linux.
- Automatically downloads and compiles Python 2.7 to 3.7+ from python.org source.
- Manages the installation of Linux OS dependencies to enable python3 advanced runtime features.
- Optional `quiet` mode for fully automated, unattended installs.

System-level Python3

- buildpy compiles, installs, and configures Python binaries for the entire system.
- If you wish to build Python for a single user, consider pipenv

Back to Table Of Contents

CHAPTER 2

Getting Started

Before starting, we recommended reviewing the following:

- Read the Frequently Asked Questions.
- Reviewing the buildpy feature overview (below) to learn about buildpy.

Back to Table Of Contents

CHAPTER 3

Documentation

Online

- Complete html documentation available at http://buildpy.readthedocs.io.

Download: Available via download in the formats below

- pdf format
- Amazon Kindle (epub) format

Source Code

- buildpython3 git repository

Back to Table Of Contents

CHAPTER 4

Supported Linux Distributions

Ubuntu, Ubuntu-based Variants

- : Ubuntu variants based on 14.04+
- : Ubuntu variants based on 16.04, 16.10+
- : Ubuntu variants based on 18.04, 18.10+
- , ,

Redhat, Redhat-based Variants

-
-
-
- (2017+)
- (2018+)

Note: Older versions than listed above may be compatible, but not have not been tested

Back to Table Of Contents

CHAPTER 5

Help

To display the help menu:

```
$ buildpy --help
```

```
                Build Python3 from Source | Linux

DESCRIPTION

      Utility to compile and install the latest versions of python3
      official source binaries download from https://www.python.org

SYNOPSIS

                $  buildpy   < OPTION >

                        -I | --install <value>
                       [-d | --download <value> ]
                       [-c | --clean <value> ]
                       [-o | --optimization ]
                       [-s | --show  <value> ]
                       [-i | --info  ]
                       [-q | --quiet ]
                       [-t | --os-detect  ]
                       [-b | --backup-pip ]
                       [-V | --version ]
                       [-U | --uninstall  <value> ]

                Note:  Requires sudo or root privileges
OPTIONS

      -b, --backup-pip: Create local Backup copy of installed python
          packages (single operation only)

      -c, --clean: Remove all build and installation artifacts. Can
          optionally be invoked with a Python major revision number
          provided as a parameter to clean specific build artifacts

      -d, --download:  Download Python build artifacts, but take no
          other action. Optionally can be invoked with Python major
          revision number to download a specific Python binary set:

                      $  buildpy  --download  3.4

      -I, --install <value>: Build, compile, and install Python bin-
          aries  on  local  machine  for  all  system users.  Value
          indicates the latest Python minor version for the release
          specified by <value> (DEFAULT: 3.6)

      -o, --optimization: Compile with Optimizations (Optional)
          Compiling with optimization drastically lengthens compile
          time; but results in increased speed of the CPython comp-
          iler (~ 10%).  Recommended only if abundant cpu resources

      -i, --info:  Display detailed information on the buildpy
          prgram and functions it contains

      -q, --quiet:  (Optional) Supress output to stdout.  Option is
          invoked for scripted or unattended compile and install.

      -s, --show: (Optional) Show latest Python minor version avail-
          able for named Python major version (DEFAULT: 3.6)

      -t, --os-detect: Detect Operating System type and exit (single
          operation only)

      -U, --uninstall:  Uninstall  the Linux system Python3 binaries
          and related libraries and support artifacts from the local
          file system for Python major revision given as a parameter

      -V, --version: Display buildpy version and license information

EXAMPLES

          Compile and install latest Python 3.6 source binary
                  $ buildpy  --install 3.6

          Compile with Optimizations, then install latest Python 3.7
                  $ buildpy  --install 3.7 --optimizations

          Show most recent Python 3.4 source binary available
                  $ buildpy  --show 3.4
```

Back to Table Of Contents

CHAPTER 6

Author & Copyright

All works contained herein copyrighted via below author unless work is explicitly noted by an alternate author.

- Copyright 2017-2021 Blake Huber, All Rights Reserved.

Sofware License

- Software is licensed and protected under the GNU General Public License Agreement v3.

Back to Table Of Contents

CHAPTER 7

Disclaimer

Code is provided "as is". No liability is assumed by either the code's originating author nor this repo's owner for their use at AWS or any other facility. Furthermore, running function code at AWS may incur monetary charges; in some cases, charges may be substantial. Charges are the sole responsibility of the account holder executing code obtained from this library.

Additional terms may be found in the complete license agreement.

Table Of Contents

CHAPTER 8

License

```
GNU GENERAL PUBLIC LICENSE
   Version 3, 29 June 2007
```

Copyright (C) 2007 Free Software Foundation, Inc. http://fsf.org/ Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

```
Preamble
```

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

```
TERMS AND CONDITIONS
```

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

```
a) The work must carry prominent notices stating that you modified
it, and giving a relevant date.

b) The work must carry prominent notices stating that it is
released under this License and any conditions added under section
```

(continues on next page)

(continued from previous page)

```
7.  This requirement modifies the requirement in section 4 to
"keep intact all notices".

c) You must license the entire work, as a whole, under this
License to anyone who comes into possession of a copy.  This
License will therefore apply, along with any applicable section 7
additional terms, to the whole of the work, and all its parts,
regardless of how they are packaged.  This License gives no
permission to license the work in any other way, but it does not
invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display
Appropriate Legal Notices; however, if the Program has interactive
interfaces that do not display Appropriate Legal Notices, your
work need not make them do so.
```

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

```
a) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by the
Corresponding Source fixed on a durable physical medium
customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by a
written offer, valid for at least three years and valid for as
long as you offer spare parts or customer support for that product
model, to give anyone who possesses the object code either (1) a
copy of the Corresponding Source for all the software in the
product that is covered by this License, on a durable physical
medium customarily used for software interchange, for a price no
more than your reasonable cost of physically performing this
conveying of source, or (2) access to copy the
Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the
written offer to provide the Corresponding Source.  This
alternative is allowed only occasionally and noncommercially, and
only if you received the object code with such an offer, in accord
with subsection 6b.

d) Convey the object code by offering access from a designated
place (gratis or for a charge), and offer equivalent access to the
Corresponding Source in the same way through the same place at no
further charge.  You need not require recipients to copy the
Corresponding Source along with the object code.  If the place to
copy the object code is a network server, the Corresponding Source
may be on a different server (operated by you or a third party)
```

(continues on next page)

(continued from previous page)

```
that supports equivalent copying facilities, provided you maintain
clear directions next to the object code saying where to find the
Corresponding Source.  Regardless of what server hosts the
Corresponding Source, you remain obligated to ensure that it is
available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided
you inform other peers where the object code and Corresponding
Source of the work are being offered to the general public at no
charge under subsection 6d.
```

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

```
a) Disclaiming warranty or limiting liability differently from the
terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or
author attributions in that material or in the Appropriate Legal
Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or
requiring that modified versions of such material be marked in
reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or
authors of the material; or

e) Declining to grant rights under trademark law for use of some
trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that
material by anyone who conveys the material (or modified versions of
it) with contractual assumptions of liability to the recipient, for
any liability that these contractual assumptions directly impose on
those licensors and authors.
```

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise

does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work

from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE

WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

```
         END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs
```

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
{one line to give the program's name and a brief idea of what it does.}
Copyright (C) {year}  {name of author}

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
{project}  Copyright (C) {year}  {fullname}
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w'` andshow c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html.

Table Of Contents

CHAPTER 9

Dependencies

- *Root Privileges*. Ability to assume user with root privileges or access such privileges via **sudo**
- **Python3-support via Operating System Packages**. Dependencies on various operating system packages available from official Linux distribution's repository. You can view os package dependencies with the following command once **buildpy** is installed:
- **Bash 4.1 or higher**. No preexisting python implementation is required. **buildpy** is written in bash.

Back to Table Of Contents

CHAPTER 10

Installation

10.1 Debian, Ubuntu, Linux Mint, Ubuntu Variants

Method 1: Install the **debian-tools** repository.

The easiest way to install **buildpy** is via the Debian-tools repository:

1. Download the public key:

```
$ wget -qO - http://awscloud.center/keys/public.key | sudo apt-key add -
```

2. Install the repository:

```
$ string='deb [arch=amd64] http://deb.awscloud.center <distribution> main' ; \
  sudo echo $string > /etc/apt/sources.list.d/debian-tools.list
```

   Where `<distribution>` is one of the following:

   - `trusty`: Ubuntu 14.04, Ubuntu 14.04 based Linux distributions
   - `xenial`: Ubuntu 16.04, 16.04 based Linux distributions
   - `bionic`: Ubuntu 18.04, 18.04 based Linux distributions

3. Verify package repository installation

```
$ apt list buildpy -a
```

4. Update and install the package:

```
$ sudo apt update  &&  sudo apt install buildpy
```

5. Verify Installation. To verify a Debian (.deb) package installation:

```
$ apt show buildpy
```

```
blake@ubuntu1-desktop:~$ echo -e "\n"; apt list buildpy -a


Listing... Done
buildpy/bionic 1.7.5 amd64
buildpy/bionic 1.7.3 amd64
buildpy/bionic 1.7.2 amd64
buildpy/bionic 1.7.1 amd64
buildpy/bionic 1.7.0 amd64
buildpy/bionic 1.6.17 amd64
buildpy/bionic 1.6.16 amd64
buildpy/bionic 1.6.15 amd64
buildpy/bionic 1.6.14 amd64
buildpy/bionic 1.6.13 amd64
```

```
Package: buildpy
New: yes
State: installed
Automatically installed: no
Version: 1.6.15
Priority: optional
Section: devel
Maintainer: Blake Huber <blakeca00@gmail.com>
Architecture: amd64
Uncompressed Size: 118 k
Depends: bash (>= 4.2), curl (>= 7.0), bc (>= 1.0), debianutils, sudo,
         util-linux
Description: Compile and Build Any Python3 Version on Linux

 buildpy is a Utility for building python3 from source binaries on Linux.

 Compile and install Python 3.0 to 3.7+ major versions. Python 2.x can be
 compiled and installed; however, this has not been tested extensively. When a
 Python major version number is specified (Python 3.6 for example), buildpy
 automatically finds and installs the latest minor version available for
 download from https://www.python.org/ftp/python. buildpy has a "quiet mode" for
 scripted configuration management installs.

 The buildpy utility builds Python binaries for the entire system. If you wish
 to build Python for a single user, consider pipenv
Homepage: https://bitbucket.org/blakeca00/buildpython3
```

Back to Table Of Contents

Method 2: Install directly via `.deb` package.

If you do not want to install an additional repository on your local machine, you may instead download and install the .deb package using the `apt` or `apt-get` package mangers.

1. **Download**: the .deb installation package (v1.6.3 shown):

```
$ wget https://bitbucket.org/blakeca00/buildpython3/downloads/buildpy-1.6.3_amd64.
↪deb
```

2. **Install** via apt package manager:

```
$ sudo apt install ./buildpy-1.6.3_amd64.deb
```

Note:

- Installation of the .deb package will also install the **debian-tools** repository on your local machine. If you do not want this, please comment out the single line in `/etc/apt/sources.list.d/debian-tools.list`.

Back to Table Of Contents

10.2 Redhat, CentOS, Fedora RPM-based Distributions

The easiest way to install **buildpy** on redhat-based Linux distributions is via the developer-tools package repository:

(**Fedora** Users: use `dnf` in place of `yum` below)

1. Download and install the repo definition file

```
$ sudo yum install wget
```

```
$ wget https://bitbucket.org/blakeca00/buildpython3/downloads/developer-tools.repo
```

```
$ sudo mv developer-tools.repo /etc/yum.repos.d/  &&  sudo chown 0:0 developer-
↪tools.repo
```

2. Update local repository cache

```
$ sudo yum update -y
```

3. Install **buildpy** os package

```
$ sudo yum install buildpy
```

   Answer "y":

4. Verify Installation

```
$ yum info buildpy
```

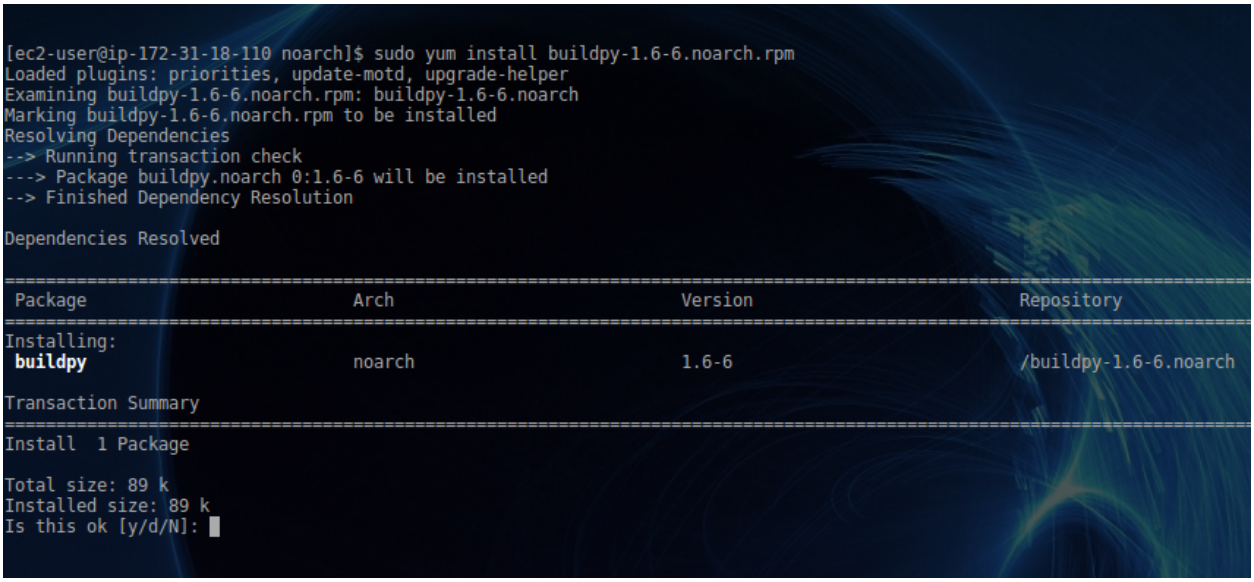


Fig. 1: rpm install

Fig. 2: rpm install

Table Of Contents

CHAPTER 11

Upgrading

To discover and apply newly-released versions of BUILDPY, follow the steps below for your specific distribution.

11.1 Debian, Ubuntu Variants

(1) Find out if an upgraded version of buildpy is available:

```
$ sudo apt update && sudo apt list --upgradeable
```

```
Listing... Done
buildpy/bionic 1.7.5 amd64 [upgradable from: 1.7.4]
N: There are 10 additional versions. Please use the '-a' switch to see them.
```

Alternate:

```
$ apt list buildpy -a
```

```
Listing... Done
buildpy/bionic 1.7.5 amd64 [upgradable from: 1.7.4]
buildpy/now 1.7.4 amd64 [installed,upgradable to: 1.7.5]
buildpy/bionic 1.7.3 amd64
buildpy/bionic 1.7.2 amd64
buildpy/bionic 1.7.1 amd64
buildpy/bionic 1.7.0 amd64
buildpy/bionic 1.6.17 amd64
buildpy/bionic 1.6.16 amd64
buildpy/bionic 1.6.15 amd64
buildpy/bionic 1.6.14 amd64
buildpy/bionic 1.6.13 amd64
```

(2) Install available upgrades:

```
$ sudo apt upgrade
```

(3) Verify upgrade:

```
$ sudo apt list buildpy -a
```

```
Listing... Done
buildpy/bionic,now 1.7.5 amd64 [installed]
buildpy/bionic 1.7.3 amd64
buildpy/bionic 1.7.2 amd64
buildpy/bionic 1.7.1 amd64
buildpy/bionic 1.7.0 amd64
buildpy/bionic 1.6.17 amd64
buildpy/bionic 1.6.16 amd64
buildpy/bionic 1.6.15 amd64
buildpy/bionic 1.6.14 amd64
buildpy/bionic 1.6.13 amd64
```

Back to Table Of Contents

11.2 Upgrading Redhat-based Distributions

(1) Find out if an upgraded version of buildpy is available

```
$ sudo yum info buildpy
```

```
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
 * base: repos-tx.psychz.net
 * extras: repos-tx.psychz.net
 * updates: repos-tx.psychz.net
Installed Packages
Name        : buildpy
Arch        : noarch
Version     : 1.7
Release     : 1
Size        : 124 k
Repo        : installed
From repo   : developer-tools
Summary     : A Utility for Compiling and Installing Python3 on Linux
URL         : https://bitbucket.org/blakeca00/buildpython3
License     : GPL
Description : Utility for compiling and installing any Python version
            : from source.
            :
            : Supports Amazon Linux v1, Amazon Linux v2, CentOS 6,
            : CentOS 7, Redhat Enterprise Linux 6 and 7

Available Packages
Name        : buildpy
Arch        : noarch
Version     : 1.7
Release     : 5
Size        : 37 k
Repo        : developer-tools/x86_64
Summary     : A Utility for Compiling and Installing Python3 on Linux
URL         : https://bitbucket.org/blakeca00/buildpython3
License     : GPL
Description : Utility for compiling and installing any Python version
            : from source.
            :
            : Supports Amazon Linux v1, Amazon Linux v2 (2018+),
            : CentOS 7, Redhat Enterprise Linux 6 and 7
```

(2) Install available upgrades:

```
$ sudo yum update -y
```

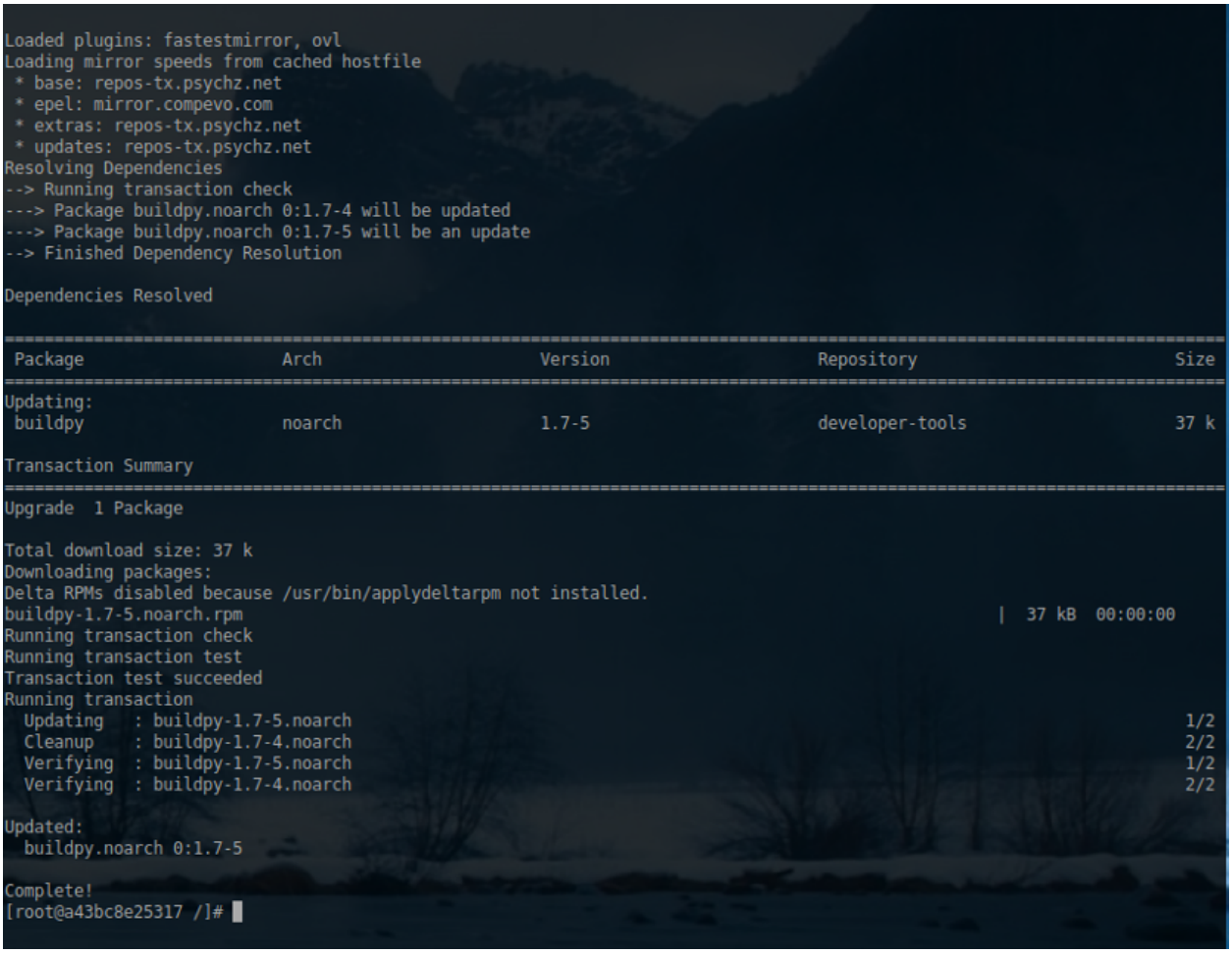


Table Of Contents

CHAPTER 12

Uninstall

12.1 Debian and Ubuntu Variants

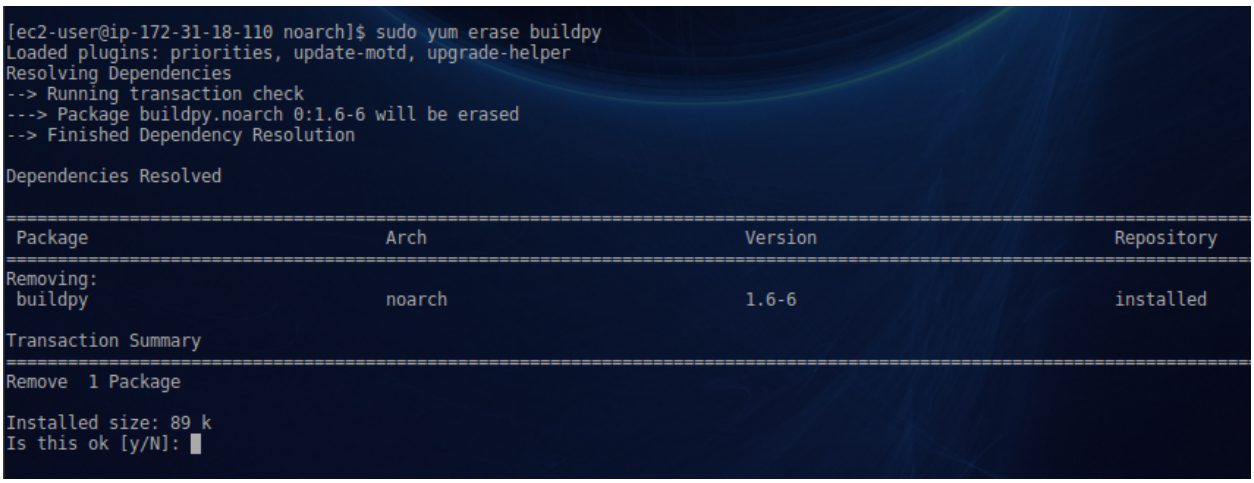
buildpy may be removed from your system using the debian package manager:

```
$ sudo apt remove buildpy
```

12.2 Redhat, CentOS, Fedora RPM Variants

buildpy may be removed from your system using the yum package manager:

```
$ sudo yum erase buildpy
```



Answer “y”:

Table Of Contents

CHAPTER 13

Use Cases

- *Verify Operating System Dependencies*
- *Compile Python 3 - Basic Installation*
- *Compile Python 3 - Advanced Installation*
- *Compile Python 3 - Specific Version*
- *Unattended Install*
- *Uninstalling Python 3 Versions*

Important:

Root privileges via sudo *required*; otherwise execute directly as root user

13.1 Verify Operating System Dependencies

```
$ buildpy --show os-packages
```

Back to *Use Cases* top

13.2 Compile Python 3 - Basic Installation

To compile and install the latest Python-3.6 binaries (version 3.6.7 at the time of this post), use the following command:

```
$ sudo buildpy --install Python-3.6
```

Or as root directly:

```
$ sudo su -
```

```
root@dev:~# buildpy --install 3.6
```

Back to *Use Cases* top

13.3 Compile Python 3 - Advanced Installation

Python 3 Installation Compiled with Optimizations

buildpy can compile Python 3 binaries with optimizations. Although this will lengthen the compile time considerably (30 minutes - 1 hour depending upon system resources), the speed of the Python interpreter code execution will increase by 10% - 20%.

```
$ sudo buildpy --install Python-3.6 --optimizations
```

Compilation Parallelism

When compiling with optimizations, you may wish to increase the number of parallel compilation processes to decrease the duration of the compile time. With **buildpy**, this is configurable via the `--parallel-processes` option. Up to 9 cpus can be utilized in parallel during compile time.

```
$ sudo buildpy --install Python-3.6 --optimizations --parallel-processes 8
```

The above options will compile Python 3.6 binaries with optimizations utilising 8 cpu's in parallel.

Note

4 cpu's is the default parallelism; i.e. 4 cpus are used in parallel during the compile.

Back to *Use Cases* top

13.4 Compile Python 3 - Specific Version

If you wish to compile and install a specific Python 3 binary, meaning a specific major and minor Python 3 version, simply add the full version label instead of just providing the Python major revision number:

```
$ sudo buildpy --install Python-3.7.6
```

Alternatively, for all installation types listed in this guide, the *Python-* prefix may be omitted:

```
$ sudo buildpy --install 3.7.6
```

The two commands above are equivalent and will compile and install Python 3 version 3.6.9.

Back to *Use Cases* top

13.5 Unattended Install

If run via unattended script, use `--quiet` to suppress stdout messages:

```
$ sudo buildpy --install 3.6 --quiet
```

A running log is created in all execution states to `/var/log/buildpy.log`. See an example log file.

Back to *Use Cases* top

13.6 Uninstalling Python 3 Versions

At any time you may choose to uninstall a system-wide, compiled version of Python 3 using **buildpy**. The following command example uninstalls Python 3.6:

```
$ sudo buildpy --uninstall 3.6 --purge
```

If a system-wide Python 3 version pre-existed prior to installing a compiled Python 3 version, the *python3* symlink will be redirected to the previous Python 3 version once the uninstall operation completes.

Important:

For stability, *only compiled Python 3 versions can be uninstalled* with **buildpy**. Native, system-wide Python versions installed from the operating system package repositories must uninstalled using the OS package manager application.

Back to *Use Cases* top

Back to Table Of Contents

CHAPTER 14

Frequently Asked Questions

General Questions

- *Q: What is buildpy? Why do I care?*
- *Q: What the main use cases for buildpy?*

Required User Permissions

- *Q: What permissions are required to compile and install python3 with buildpy?*
- *Q: What happens if I run buildpy as root?*

Python Version Support

- *Q: What python3 versions can I compile and install with buildpy*
- *Q: Why do I not just install python3 from my os distribution's package repository insted of using buildpy?*
- *Q: Does buildpy overwrite the old system-level Python3 version?*

Supported Linux Operating System Distributions

- *Q: If buildpy is written in bash, I can run it on any Linux OS, right?*
- *Q: What Redhat-based distributions does buildpy install via an RPM package?*

Uninstall Python3

- *Q: What Python3 versions can I safely remove from my system with buildpy?*

Logging FAQ

- *Q: Which log files does buildpy write to?*

Miscellaneous Questions

- *Q: Since buildpy requires an Internet connection to download python binaries, does it send any information to you or anyone else?*

14.1 General Questions

14.1.1 Q: What is buildpy? Why do I care?

A: Seriously, you may not care, but if you use Linux, you probably should. buildpy is only a fantastic utility or tool for anyone who needs to install a specific version of Python on a Linux hardware, virtual machine, or container. It is also something anyone who has to remove and replace a version of Python in machine or machine image.

14.1.2 *Q*: What the main use cases for buildpy?

A: General use cases for buildpy are:

1. Use buildpy to find out when new python X.Y version is available (below)

```
$ buildpy --show os-packages
```

2. You want an easy to use, reliable means of compiling the latest version of Python after release from https://python.org.

```
$ buildpy --profile default --operation list        # list key information
```

3. _Uninstall_: You need to install a version of Python3 you compiled on your machine (not a system Python that came with your distribution)

Back to *Frequently Asked Questions* top

14.2 Required User Permissions

14.2.1 Q: What permissions are required to compile and install python3 with buildpy?

A: It depends on what functionality you wish to invoke. buildpy runs the majority of operations under normal user permissions. The exception is when compiling and installing a new version of Python3. This is because buildpy installs a compiled version of Python3 for the entire system; thus, the `--install` operation requires root level permissions via sudo or assumed by running buildpy as root directly.

The following requires sudo or root permissions:

```
$ sudo  buildpy  --install  Python3.7  [ --optimizations ]
```

14.2.2 Q: What happens if I run buildpy as root?

A: In essence, nothing changes when you run buildpy directly as root. You are installing a new version of Python3 for the entire system either when running buildpy as sudo or directly via root permissions.

Back to *Frequently Asked Questions* top

14.3 Python Version Support

14.3.1 Q: What python3 versions can I compile and install with buildpy

A: It depends on what functionality you wish to invoke. buildpy runs the majority of operations under normal user permissions. The exception is when compiling and installing a new version of Python3. This is because buildpy installs a compiled version of Python3 for the entire system; thus, the `--install` operation requires root level permissions via sudo or assumed by running buildpy as root directly.

The following requires sudo or root permissions:

```
$ buildpy  --install  Python3.7  [ --optimizations ]
```

Back to *Frequently Asked Questions* top

14.3.2 Q: Why do I not just install python3 from my os distribution's package repository insted of using buildpy?

A: That is a bit of a philosophical question, so let's just stick to hard benefits that are widely accepted. The reason you would use buildpy to compile a custom version of python3 are:

1. Compiled Python interpreters are reported to enjoy an approximate 10% execution speed improvement
2. You care about upgrading to the latest minor Python3.7 revision when it is released (example: 3.7.2 to 3.7.3).
3. You are required to run an *older* Python version (say, Python 3.4) and a modern machine for testing purposes
4. You want to enable custom functionality not enabled in the Python3 binaries installed from your OS distribution's package repository.

Back to *Frequently Asked Questions* top

14.3.3 Q: Does buildpy overwrite the old system-level Python3 version?

A: No. Let's say you have Python3.4 installed by default on a Ubuntu 14.04 system. You want to install the latest . At the completion of the installation, will still function for os-dependent subsystems.

Back to *Frequently Asked Questions* top

14.4 Supported Linux Operating System Distributions

14.4.1 Q: If buildpy is written in bash, I can run it on any Linux OS, right?

A: Yes and No actually, or as they say in situation like this: *it depends*.

It depends upon meeting the minimum requirements outlined in the depenencies section. `Bash 4.4+` is required as well as a few other critical os packages such as , , and .

Back to *Frequently Asked Questions* top

14.4.2 Q: What Redhat-based distributions does buildpy install via an RPM package?

A: See the list . buildpy is also compatible with Amazon Linux 2 when run on EC2 virtual machines.

Back to *Frequently Asked Questions* top

14.5 Uninstall Python3

14.5.1 Q: What Python3 versions can I safely remove from my system with buildpy?

A: As a built-in safeguard, buildpy will only remove compiled versions from your system. This is any python installed in `/usr/local/`. This is location of any python versions compiled from source.

Most if not all system-wide Python binaries install from your Linux distribution's official package repository reside in `/usr/bin`; thus, buildpy avoids this location and will *not remove* python binaries from this location as a precaution.

To initiate the uninstall operation, type the following:

```
$ sudo buildpy --uninstall Python3.6
```

Back to *Frequently Asked Questions* top

14.6 Logging FAQ

14.6.1 Q: Which log files does buildpy write to?

A: buildpy produces the following log files:

- `/var/log/buildpy.log` contains summary messages produced during normal operations
- `/var/log/console.log` contains all console messages produced only during compile and install operations

Back to *Frequently Asked Questions* top

14.7 Miscellaneous Questions

14.7.1 Q: Since buildpy requires an Internet connection to download python binaries, does it send any information to you or anyone else?

A: None whatsoever. buildpy can actually function fine without any Internet connection if the python binaries are downloaded offline and placed in `/tmp` so that they used as the source during the install.

Back to *Frequently Asked Questions* top

Table Of Contents

CHAPTER 15

Screenshot Index

System Info (`--info`)

- *Operating System Detection*
- *Program Information*

Show Functionality (`--show`):

- *Python Binary Version Status*
- *OS Packages Installed Status*

Download (`--download`):

- *Download Python Source (Default)*
- *Download Python Source, Named Version*

Logging

- *Log Files*

Clean (`--clean`):

- *Remove Build Artifacts*

Back to Table Of Contents

CHAPTER 16

Operating System Detection

Test OS detection

```
$ buildpy  --os-detect
```

Ubuntu / Linux Mint

Amazon Linux

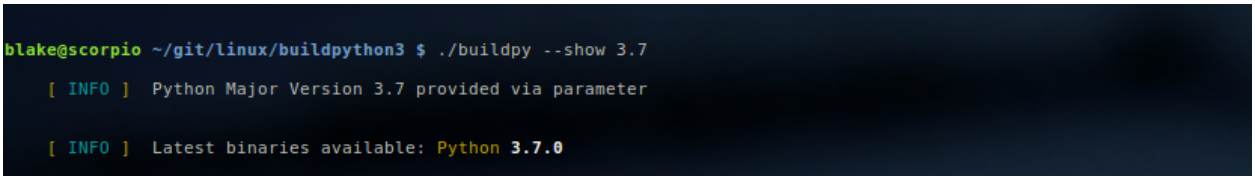
Back to *Screenshot Index* index

CHAPTER 17

Python Binary Version Status

Show the latest `Python 3.7` version available for install

```
$ buildpy  --show  Python-3.7
```



Back to *Screenshot Index* index

CHAPTER 18

OS Packages Installed Status

Show operation system package prerequisites and installed status of each

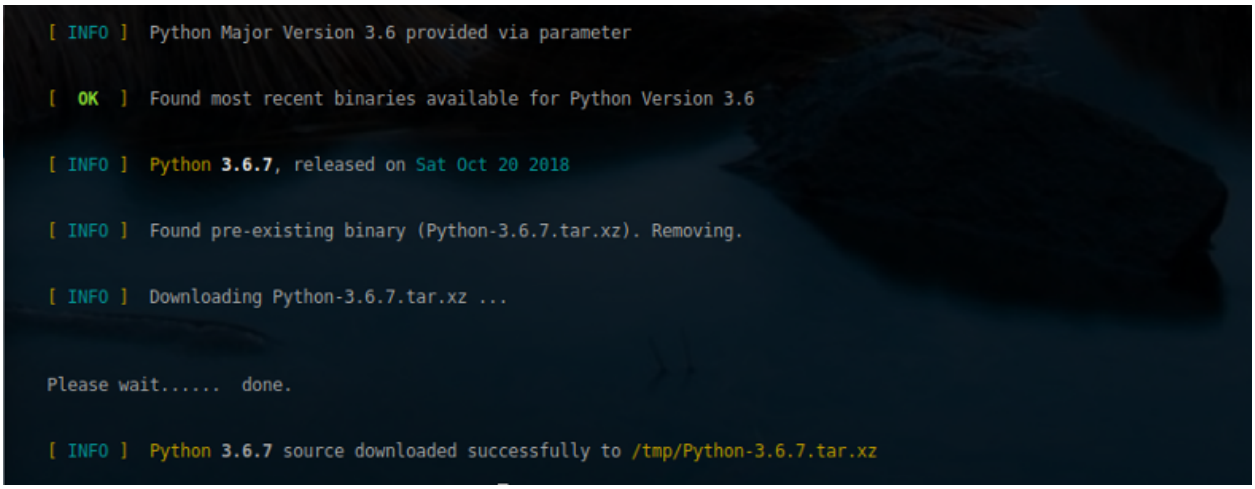
```
$ buildpy  --show  os-packages
```

Back to *Screenshot Index* index

CHAPTER 19

Download Python Source (Default)

```
$ buildpy  --download        # default, download Python 3.6 binaries
```



Back to *Screenshot Index* index

CHAPTER 20

Download Python Source, Named Version

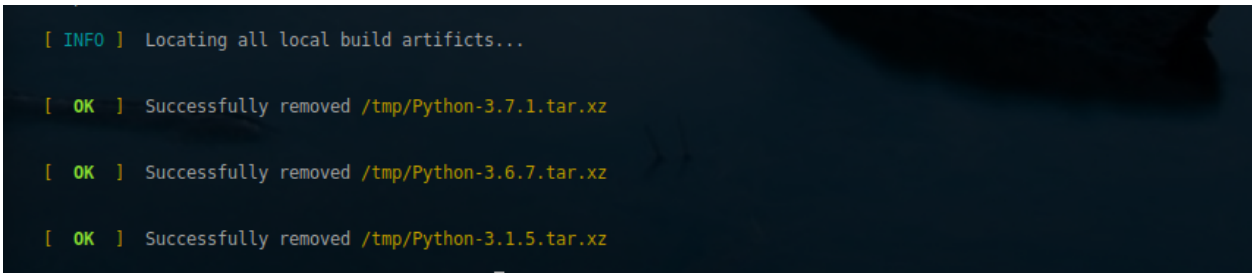
```
$ buildpy  --download  3.4
```

Back to *Screenshot Index* index

CHAPTER 21

Remove Build Artifacts

```
$ buildpy --clean
```



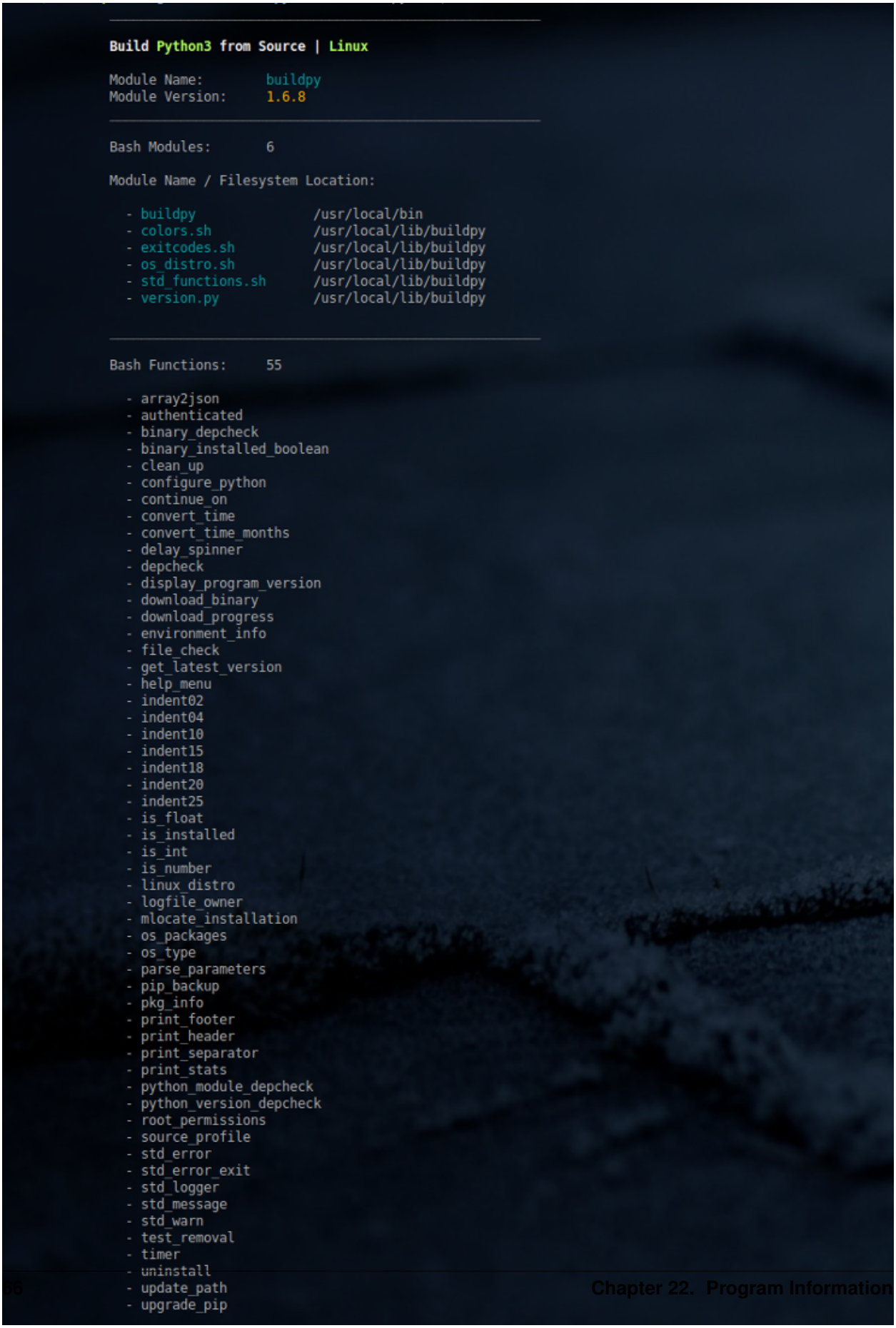
Back to *Screenshot Index* index

CHAPTER 22

Program Information

Detailed information regarding the local installation of buildpy program and dependencies:

```
$ buildpy --info
```


Build Python3 from Source | Linux
Module Name: buildpy
Module Version: 1.6.8
Bash Modules: 6
Module Name / Filesystem Location:
- buildpy /usr/local/bin
- colors.sh /usr/local/lib/buildpy
- exitcodes.sh /usr/local/lib/buildpy
- os_distro.sh /usr/local/lib/buildpy
- std_functions.sh /usr/local/lib/buildpy
- version.py /usr/local/lib/buildpy
Bash Functions: 55
- array2json
- authenticated
- binary_depcheck
- binary_installed_boolean
- clean_up
- configure_python
- continue_on
- convert_time
- convert_time_months
- delay_spinner
- depcheck
- display_program_version
- download_binary
- download_progress
- environment_info
- file_check
- get_latest_version
- help_menu
- indent02
- indent04
- indent10
- indent15
- indent18
- indent20
- indent25
- is_float
- is_installed
- is_int
- is_number
- linux_distro
- logfile_owner
- mlocate_installation
- os_packages
- os_type
- parse_parameters
- pip_backup
- pkg_info
- print_footer
- print_header
- print_separator
- print_stats
- python_module_depcheck
- python_version_depcheck
- root_permissions
- source_profile
- std_error
- std_error_exit
- std_logger
- std_message
- std_warn
- test_removal
- timer
- uninstall
- update_path
- upgrade_pip

Back to *Screenshot Index* index

CHAPTER 23

Log Files

buildpy actively maintains 2 log files:

1) **buildpy.log**
    - Logging for all non-compile functions
    - Main system log messages (shown below). Located at `/var/log/buildpy.log`.
2) **console.log**:
    - Messages during Compile & Build
    - Populated for tracking purposes when `--quiet` flag set to surpress output to stdout.

```
$ tail -n 100 /var/log/buildpy.log
```

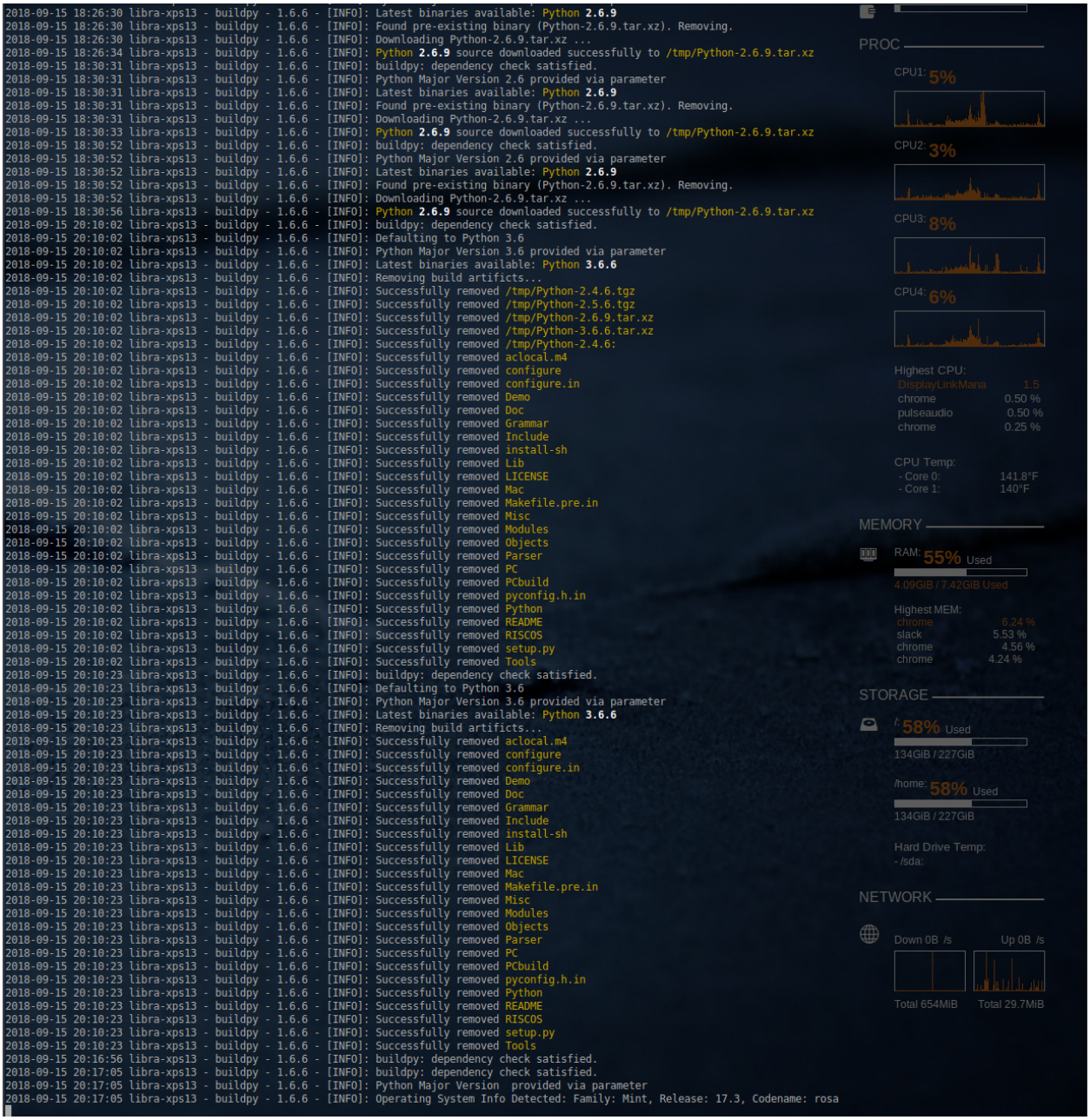


Table Of Contents

CHAPTER 24

buildpy Source Code

24.1 Bash Module Index

- *buildpy*
- *std_functions.sh*
- *os_distro.sh*
- *colors.sh*
- *exitcodes.sh*
- *version.py*

Table Of Contents

24.2 buildpy

Back to *Bash Module Index*

```
#!/usr/bin/env bash

#
#   Summary:
#       - Script to retrieve, compile, and install specific python3 interpreter
#       - Compile and install Python3 versions: 3.0 thru 3.7+
#       - Binaries installed at OS level enable ipython, sqlite, other useful
```

(continues on next page)

(continued from previous page)

```
#          packages during python development
#
#
#   Copyright 2017-2018 Blake Huber.  All rights reserved.
#


# --- global vars ------------------------------------------------------------
--------------


pkg=$(basename $0)                                      # pkg (script) full name
pkg_root=$(echo $pkg | awk -F '.' '{print $1}')         # pkg without file extention
pkg_path=$(cd $(dirname $0); pwd -P)                    # location of pkg
pkg_lib="$pkg_path/core"
TMPDIR='/tmp'
workdir="$HOME/Downloads"
NOW=$(date +'%Y%m%d')
LOG_DIR="$HOME/logs"
LOG_FILE="$LOG_DIR/$pkg_root.log"
LOG_CONSOLE="$LOG_DIR/console.log"

# source colors, exitcodes, version file, and  std_function defs | NOTE: source order␣
↪dependent
source $pkg_lib/colors.sh
source $pkg_lib/exitcodes.sh
source $pkg_lib/version.py
source $pkg_lib/std_functions.sh

# formatting
bbc=$(echo -e ${bold}${a_brightcyan})                   # bold bright cyan highlight
bbw=$(echo -e ${bold}${a_brightwhite})                  # bold, bright white highlight
title=$(echo -e ${bold}${a_brightwhite})                # title color, white + bold
hic=$(echo -e ${bold}${a_brightyellowgreen})            # help menu accent 1
bin=$(echo -e ${bold}${a_orange})                       # help menu binary accent
ul=$(echo -e ${underline})                              # std underline
bd=$(echo -e ${bold})                                   # std bold
wt=$(echo -e ${a_brightwhite})                          # help menu accent 2
fs=$(echo -e ${yellow})                                 # file path color
btext=${reset}                                          # clear accents; rtn to native␣
↪term colors

# optimizations increase performance, but lengthen
# build time (see -o, --optimizations parameter)
OPTIMIZATIONS="false"
PARALLEL_JOB_CT='4'
DEFAULT_MAJOR_VERSION='3.6'


# --- declarations -----------------------------------------------------------
--------------


function help_menu(){
    cat <<EOM

                    ${title}Build ${hic}Python3 ${title}from Source | ${hic}Linux$
↪{btext}
```

(continues on next page)

(continued from previous page)

```
61
62    ${title}DESCRIPTION${btext}
63
64          Utility to compile and install the latest versions of python3
65          official source binaries download from ${url}https://www.python.org${btext}
66
67    ${title}SYNOPSIS${btext}
68
69             $  ${bin}$pkg${reset}   --install  ${bbc}|${btext}  --uninstall  ${bbc}|$
      ↪{btext}  --show
70
71                            -I | --install <value>
72                            -d | --download <value>
73                           [-c | --clean <value> ]
74                           [-o | --optimization  ]
75                            -s | --show  <value>
76                           [-i | --info  ]
77                           [-q | --quiet ]
78                           [-t | --os-detect  ]
79                           [-b | --backup-pip ]
80                           [-V | --version  ]
81                            -U | --uninstall  <value>
82
83                  ${title}Note:${btext}  Requires sudo or root privileges
84
85    ${title}OPTIONS${btext}
86          ${title}-b${btext}, ${title}--backup-pip${btext}: Create local Backup copy of␣
      ↪installed python
87              packages (single operation only)
88
89          ${title}-c${btext}, ${title}--clean${btext}: Remove all build and␣
      ↪installation artifacts. Can
90              optionally be invoked with a Python major revision number
91              provided as a parameter to clean specific build artifacts
92
93          ${title}-d${btext}, ${title}--download${btext}:  Download Python build␣
      ↪artifacts, but take no
94              other action. Optionally can be invoked with Python major
95              revision number to download a specific Python binary set
96
97          ${title}-I${btext}, ${title}--install${btext} <value>: Build, compile, and␣
      ↪install Python bin-
98              aries  on  local  machine  for  all  system users.  Value
99              indicates the latest Python minor version for the release
100             specified by <value>:
101
102                        $  sudo  ${bin}$pkg${reset}  --install  3.7
103
104         ${title}-o${btext}, ${title}--optimizations${btext}:  Compile with configure␣
      ↪option --enable-
105             optimizations  turned-on.  Increases duration of comnpile
106             time,  but results in increased execution speed of the C-
107             Python compiler.  Recommended if abundant cpu resources.
108
109         ${title}-i${btext}, ${title}--info${btext}:  Display detailed information on␣
      ↪the $pkg
110             prgram and functions it contains
```

(continues on next page)

(continued from previous page)

```
111
112         ${title}-q${btext}, ${title}--quiet${btext}:  (Optional) Supress output to␣
↪stdout.  Option is
113             invoked for scripted or unattended compile and install.
114
115         ${title}-s${btext}, ${title}--show${btext}: (Optional) Show latest Python␣
↪minor version avail-
116             able for named Python major version (DEFAULT: 3.6)
117
118         ${title}-t${btext}, ${title}--os-detect${btext}: Detect Operating System type␣
↪and exit (single
119             operation only)
120
121         ${title}-U${btext}, ${title}--uninstall${btext}:  Uninstall  the Linux system␣
↪Python3 binaries
122             and related libraries and support artifacts from the local
123             file system for Python major revision given as a parameter
124
125         ${title}-V${btext}, ${title}--version${btext}: Display $pkg version and␣
↪license information
126
127   ${title}EXAMPLES${btext}
128             ${title}Compile and install latest Python3.7 with Optimizations${btext}
129                 $ sudo $pkg  --install 3.7 --optimizations
130
131             ${title}Show which os-package dependencies are available${btext}
132                 $ sudo $pkg  --show os-packages
133
134             ${title}Show most recent Python 3.4 source binary available${btext}
135                 $ $pkg  --show Python-3.4
136   ____________________________________________________________________________
137
138             ${btext}Documentation:  ${url}https://buildpy.readthedocs.io${btext}
139   ____________________________________________________________________________
140 ${reset}
141 EOM
142     #
143     # <-- end function put_rule_help -->
144 }
145
146
147 function create_global_symlink(){
148     ##
149     ##  Creates python3 symlink to python executable if missing
150     ##
151     local major_version="$1"
152     local log_file="$2"
153
154     if [ ! "$(command -v python3 2> /dev/null )" ]; then
155         ln -s /usr/local/bin/"$major_version" /usr/bin/python3
156         std_message "Created global symlink to Python $major_version binary" "INFO" "
↪$log_file"
157     else
158         std_message "Found global symlink to Python $major_version. Skip creation."
↪"INFO" "$log_file"
159     fi
160     #
```

(continues on next page)

(continued from previous page)

```
    #<-- end function create_global_symlink -->
}


function install_subcommand_help(){
    ##
    ##   Display help menu for install subcommands
    ##
    cat <<EOM

                        ${title}install${btext} options help

  ${title}DESCRIPTION${btext}

        Optional parameters for use when installing Python3 binaries
        Complementary options to the --install command-line option.

  ${title}SYNOPSIS${btext}

            $ ${bin}$pkg${reset}  --install  <${bbc}value${btext}>   [ OPTIONS ]

                         [ --optimizations  ]
                         [ --parallel-processes  ]
                         [ --quiet  ]

  ${title}OPTIONS${btext}

        ${title}-I${btext}, ${title}--install${btext} <value>: Build, compile, and install Python bin-
            aries  on  local  machine  for  all  system users.  Value
            indicates the latest Python minor version for the release
            specified by <value>:

                    $  sudo  ${bin}$pkg${reset}  --install  3.7

        ${title}-p, --parallel-processes${btext} <value>: Number of parallel processes
            (1-9) to use during compile with make.  Setting to a value
            of one (1) equals no parallelism, Default = 4.  Use of low
            setting on systems with low amount of free resources rec-
            commended.

        ${title}-o${btext}, ${title}--optimizations${btext}:  Compile with configure option --enable-
            optimizations  turned-on.  Increases duration of comnpile
            time,  but results in an increased execution speed of the
            C-Python compiler. Recommended if abundant cpu resources.

        ${title}-q${btext}, ${title}--quiet${btext}:  (Optional) Supress output to stdout.  Option is
            invoked for scripted or unattended compile and install.

        ${title}help${btext}:  Option to display the commit-log help menu contents

                $ ${BOLD}$pkg${reset}  --install  ${cyan}help${btext}
        ____________________________________________________________
    ${reset}
EOM
```

(continues on next page)

(continued from previous page)

```
    #
    # <-- end function commit_log_help -->
}


function is_float(){
    local num="$1"
    local regex='^[0-9]+[.][0-9]+?$'
    local regex2='^[0-9]+[.][0-9][0-9]+?$'
    #
    if [[ $num =~ $regex ]] || [[ $num =~ $regex2 ]]; then
        # int or float
        return 0
    fi
    # not a number
    return 1
}


function parse_parameters() {
    ##
    ##  Parse all command-line parameters
    ##

    local var parameter major

    if [[ ! $@ ]]; then
        help_menu | more
        exit 0
    else
        while [ $# -gt 0 ]; do
            case $1 in
                -h | --help)
                    help_menu | more
                    shift 1
                    exit 0
                    ;;

                -b | --backup-pip)
                    PIP_BACKUP="true"
                    shift 1
                    ;;

                -c | --clean)
                    CLEAN_UP="true"
                    if [ "$2" ]; then
                        case "$2" in
                            'ALL' | 'all' | 'everything')
                                CLEAN_VERSION="ALL"
                                shift 2
                                ;;
                            *)
                                var="$2"
                                parameter=${var#Python-}
                                if is_float "$parameter"; then
                                    MAJOR_VERSION="$parameter"
                                    CLEAN_VERSION="$parameter"
```

(continues on next page)

(continued from previous page)

```
                                shift 2
                            else
                                std_error_exit "You must provide Python major version (Example: 3.7)"
                            fi
                            ;;
                    esac

                else
                    CLEAN_VERSION="ALL"
                    shift 1
                fi
                shift 1
                ;;

            '-P' | '--purge')
                # uninstall option, full system expunge
                WIPE_CLEAN=true     # clean invoked with --uninstall option
                shift 1
                ;;

            -d | --download)
                DOWNLOAD_ONLY="true"
                if [ "$2" ]; then
                    var="$2";
                    parameter=${var#Python-}
                    if is_float "$parameter"; then
                        MAJOR_VERSION="$parameter"
                        shift 2
                    else
                        std_error_exit "You must provide Python major version (Example: 3.7)"
                    fi
                else
                    MAJOR_VERSION="$DEFAULT_MAJOR_VERSION"
                    std_message "Defaulting to Python $MAJOR_VERSION" "INFO" $LOG_FILE
                    shift 1
                fi
                shift 1
                ;;

            -I | --install)
                    INSTALL="true"

                    if [ "$2" ] && [ "$2" = "help" ]; then
                        install_subcommand_help
                        exit 0

                    elif [ "$2" ] && [ "$2" = "os-packages" ]; then
                        OPERATION="INSTALL_OS_DEPS"     # install os dependencies
                        INSTALL_OS_DEPS="true"
                        shift 2

                    elif [ "$2" ]; then
                        var="$2"
                        parameter=${var#Python-}
```

(continues on next page)

(continued from previous page)

```
                            # stdout allowed
                            case "$parameter" in
                                [0-9].[0-9] | [0-9].[0-9][0-9]  | [0-9].[0-9].[0-9] |␣
↪[0-9].[0-9][0-9].[0-9])
                                        # floating point - Python major version number
                                        MAJOR_VERSION="$parameter"
                                        shift 2
                                        ;;
                                "os" | "packages" | "deps" | "os_deps" | "os_packages
↪")
                                        OPERATION="INSTALL_OS_DEPS"     # install os␣
↪dependencies
                                        INSTALL_OS_DEPS="true"
                                        shift 2
                                        ;;
                                * )
                                        tab10='          '
                                        msg2="${tab10}or 'os-packages' to install␣
↪operating system depedencies."
                                        std_message "You must provide either a Python␣
↪Major Version (Example: 3.6),\n${msg2}" "INFO" $LOG_FILE
                                        exit $E_DEPENDENCY
                                        ;;
                            esac
                        else
                            std_error_exit "You must provide Python major version␣
↪(Example: 3.7) or 'os_deps'"
                        fi
                    ;;

                -t | --os-detect)
                    OS_DETECT="true"
                    shift 1
                    ;;

                -o | --optimization | --optimizations | --optimize)
                    OPTIMIZATIONS="true"
                    shift 1
                    ;;

                -i | --info)
                    OPERATION="PKG_INFO"
                    PKG_INFO="true"
                    shift 1
                    ;;

                -p | --parallel-processes)
                    if [ "$2" ] && [ "$2" = "help" ]; then
                        install_subcommand_help
                        exit 0
                    elif [ "$2" ]; then
                        var="$2"
                        PARALLEL_JOB_CT=${var%\-*}
                        shift 2
                    else
                        std_error_exit "You must provide a parallel process count (0-
↪9) when using --parallel-processes option" $E_DEPENDENCY
```

(continues on next page)

(continued from previous page)

```
                    fi
                    ;;

                -q | --quiet)
                    QUIET="true"
                    shift 1
                    ;;

                -s | --show)
                    if [ "$2" ]; then
                        OPERATION='SHOW'
                        SUBCMD="$2"
                        if [ $3 ]; then
                            alternate_os="$3"
                            shift 3
                        else
                            shift 2
                        fi
                    else
                        shift 1
                    fi
                    ;;

                -U | --uninstall)
                    UNINSTALL="true"
                    if [ "$2" ]; then
                        if [ "$(echo "$2" | grep Python)" ]; then
                            # convert to major version number
                            var="$2"; parameter=${var#Python-}
                        else
                            parameter="$2"
                        fi
                        if is_float "$parameter"; then
                            MAJOR_VERSION="$parameter"
                            UNINSTALL_VERSION="$parameter"
                        else
                            std_error_exit "You must provide Python major version (Example: 3.7)"
                        fi
                        shift 2
                    fi
                    ;;

                -V | --version)
                    DISPLAY_VERSION="true"
                    shift 1
                    ;;

                *)
                    echo "unknown parameter. Exiting"
                    exit 1
                    ;;
            esac
        done
    fi

    # defaults
```

(continues on next page)

(continued from previous page)

```
    if [ ! $MAJOR_VERSION ]; then

        # Python Major version
        MAJOR_VERSION="$DEFAULT_MAJOR_VERSION"
        std_logger "Defaulting to Python $MAJOR_VERSION" "INFO" $LOG_FILE

    else

        # major version supplied with command options
        std_message "Python Major Version $MAJOR_VERSION provided via parameter" "INFO
↪" $LOG_FILE

    fi

    # eliminate tty output if quiet=true
    if [ $QUIET ]; then tty="/dev/null"; else tty="/dev/tty"; fi
    #
    # <-- end function parse_parameters -->
}


function binary_depcheck(){
    ##
    ## exit if binary dependencies not installed ##
    ##
    local check_list=( "$@" )
    local msg

    for prog in "${check_list[@]}"; do

        if ! type "$prog" > /dev/null 2>&1; then
            # print error msg
            msg="${title}$prog${reset} is required and not found in the PATH (code $E_
↪DEPENDENCY)"
            std_warn "$msg" $LOG_FILE
            # print pkg description info
            printf -- '%s\n' "$(cat $pkg_lib/$prog.pkg)"
            exit $E_DEPENDENCY
        fi

    done
    #
    # <<-- end function binary_depcheck -->>
}


function reset_rc_version(){
    ##
    ##  Reset VERSION global variable in case binary set to be
    ##  compiled is a release candidate, alpha, or beta code.
    ##
    local version="$1"

    if [[ $(ls $TMPDIR | grep "Python-$version.tar.xz") ]]; then
        echo "$version"
        std_logger "VERSION retained as ${fs}$VERSION${reset} -- no refactor required
↪" "INFO" $LOG_FILE
```

(continues on next page)

(continued from previous page)

```
    elif [[ $(ls $TMPDIR | grep "Python-${version}rc[1-4].tar.xz") ]]; then
        bin=$(ls $TMPDIR | grep "Python-${version}rc[1-4].tar.xz")
        echo "$(echo $bin | awk -F '-' '{print $2}')" | awk -F 'tar' '{print $1}' | rev | cut -c 2-20 | rev
        std_logger "Reset VERSION to ${fs}$VERSION${reset}" "INFO" $LOG_FILE

    elif [[ $(ls $TMPDIR | grep "Python-$version[a-b][1-4].tar.xz") ]]; then
        bin=$(ls $TMPDIR | grep "Python-${version}[a-b][1-4].tar.xz")
        echo "$(echo $bin | awk -F '-' '{print $2}')" | awk -F 'tar' '{print $1}' | rev | cut -c 2-20 | rev
        std_logger "Reset VERSION to ${fs}$VERSION${reset}" "INFO" $LOG_FILE
    fi
}


function binary_installed_bool(){
    ##
    ## return boolean value if binary dependencies installed ##
    ##
    local check_list=( "$@" )
    #
    for prog in "${check_list[@]}"; do
        if ! type "$prog" > /dev/null 2>&1; then
            return 1
        fi
    done
    return 0
    #
    # <<-- end function binary_installed_bool -->>
}


function configure_python(){
    ##
    ## function runs configure stage with appropriate options ##
    ##
    local optimizations="$1"
    local quiet="$2"
    #
    if [ "$optimizations" = "true" ] && [ ! $quiet ]; then

        ./configure --enable-optimizations  | tee /dev/tty >> $LOG_CONSOLE

    elif [ "$optimizations" = "true" ] && [ $quiet ]; then

        ./configure --enable-optimizations  >> $LOG_CONSOLE

    elif [ "$optimizations" = "false" ] && [ ! $quiet ]; then

        ./configure  | tee /dev/tty >> $LOG_CONSOLE

    elif [ "$optimizations" = "false" ] && [ $quiet ]; then

        ./configure  >> $LOG_CONSOLE

    fi
```

(continues on next page)

(continued from previous page)

```
541
542     return 0
543     #
544     # <-- end function configure_python -->
545 }
546
547
548 function continue_on(){
549     ##
550     ## user prompts to continue path ##
551     ##
552     local msg="$1"
553     local choice
554
555     if [ "$msg" ]; then
556         std_message "$msg" "INFO" $LOG_FILE
557     fi
558
559     read -p "$(printf -- '\t\tContinue? [quit]:   \n\n' )" choice
560
561     if [ -z $choice ] || [ "$choice" = "q" ] || [ "$choice" = "quit" ]; then
562         return 1
563     else
564         case $choice in
565             'Yes' | 'yes' | 'Y' | 'y' | true)
566                 return 0
567                 ;;
568             *)
569                 return 1
570         esac
571     fi
572     #
573     # <-- end function continue_on -->
574 }
575
576
577 function _current_downloads(){
578     ##
579     ##  Examines local fs for downloaded artifacts
580     ##
581     ##      - returns entry for each python binary set downloaded to /tmp
582     ##
583     local index="0"
584     local IFS
585     declare -a arr_targets xz tgz sorted_targets
586
587     # populate tar.xz, tgz arrays
588     mapfile -t xz < <(find /tmp -name \*.tar.xz 2>/dev/null)
589     mapfile -t tgz < <(find /tmp -name \*.tgz 2>/dev/null)
590
591     for i in "${xz[@]}"; do
592         temp="$(echo $i | awk -F '.tar' '{print $1}' | awk -F '.' '{print $1"."$2}')"
593         xz[$index]=$(echo $temp | awk -F '/' '{print $NF}')
594         (( index++ ))
595     done
596
597     index="0"
```

(continues on next page)

(continued from previous page)

```
    for i in "${tgz[@]}"; do
        temp="$(echo $i | awk -F '.tgz' '{print $1}' | awk -F '.' '{print $1"."$2}')"
        tgz[$index]=$(echo $temp | awk -F '/' '{print $NF}')
        (( index++ ))
    done

    arr_targets=( "$(echo "${tgz[@]}")"  "$(echo "${xz[@]}")" )
    # sort
    IFS=$'\n' sorted_targets=($(sort <<<"${arr_targets[*]}"))
    unset IFS

    echo "${sorted_targets[@]}"
    #
    # <--- end function _clean_subcommands --->
}


function depcheck(){
    ##
    ## validate cis report dependencies ##
    ##
    local log_dir="$1"
    local log_file="$2"
    local console_log="$3"
    local msg

    ## test default shell ##
    if [ ! -n "$BASH" ]; then
        # shell other than bash
        msg="Default shell appears to be something other than bash. Please rerun with bash. Aborting (code $E_BADSHELL)"
        printf -- "\n%s\n" "$msg"
    fi

    ## logging prerequisites  ##
    if [[ ! -d "$log_dir" ]]; then

        if ! mkdir -p "$log_dir"; then
            printf -- "\n%s: failed to make log directory: %s. Exit\n" % $pkg $log_dir
        fi

    elif [ ! -f $log_file ]; then

        if ! touch $log_file 2>/dev/null; then
            printf -- "\n%s: failed to seed log file: %s. Exit\n" % $pkg $log_file
        fi

    elif [ ! -f $console_log ]; then

        if ! touch $console_log 2>/dev/null; then
            printf -- "\n%s: failed to seed console log file: %s. Exit\n" % $pkg $console_log
        fi

    fi

```

(continues on next page)

(continued from previous page)

```
    ## working directory for user
    if [[ ! -d "$workdir" ]]; then

        if ! mkdir -p "$workdir"; then
            std_error_exit "$pkg: failed to make log directory: $workdir. Exit" $E_
↪DEPENDENCY
        fi

    fi

    ## check for required cli tools ##
    binary_depcheck awk bc curl

    # success
    std_logger "$pkg: dependency check satisfied." "INFO" $log_file

    # set path
    PATH=$PATH:/usr/local/bin
    return 0
    #
    # <<-- end function depcheck -->>
}


function display_program_version(){
    ##
    ## output script version info, license
    ##
    local _version=$__version__
    local _hic=$(echo -e ${a_brightblue})
    local _year=$(date +%G)
    local _bashver="$(
                bash --version | head -n1 | awk -F 'version' '{print $2}' \
                    | awk '{print $1}' | awk -F '(' '{print $1}'
            )"
    #
    cat <<EOM
    ____________________________________________________________________________




              ${_hic}$pkg${reset} version: ${title}$_version${reset}    |   GNU Bash $_
↪bashver




    ____________________________________________________________________________

      Copyright 2017-$_year, Blake Huber.  This program distributed under
      GPL v3.  This copyright notice must remain with derivative works.
    ____________________________________________________________________________

EOM
}
```

(continues on next page)

(continued from previous page)

```
708
709
710 function download_progress(){
711     ##
712     ## display progress for downloading large files, wget only
713     ##
714     local url="$1"
715
716     wget --progress=dot $url 2>&1 | grep --line-buffered "%" | \
717     sed -u -e "s,\.,,g" | awk '{printf("\b\b\b\b%4s", $2)}'
718     echo -ne "\b\b\b\b"
719     echo " Download Complete"
720     return 0
721 }
722
723
724 function download_binary(){
725     ##
726     ## download python components ##
727     ##
728     local version="$1"
729     local major="$2"
730     local minor="$(echo $version | awk -F '.' '{print $NF}')"
731     local bin_file="Python-$version.tar.xz"
732     local rc_file="Python-${version}rc[1-4].tar.xz"
733     local beta_file="Python-$version[a-b][1-4].tar.xz"
734     local bin_dcr                                   # N-1 minor release
735     local bin_alt="Python-$version.tgz"
736     local bin                                       # loop binfile
737     local url
738     local cached_download
739     declare -a arr_versions
740
741     function download_rc(){
742         ## Download release candidate binary set
743         local bin
744
745         cd $TMPDIR || exit $E_OSERROR
746
747         for rtype in rc b a; do
748             for i in 4 3 2 1; do
749                 bin="Python-${version}${rtype}${i}.tar.xz"
750
751                 # check if pre-existing; remove
752                 if [ -f "$TMPDIR/$bin" ]; then
753                     rm -f "$TMPDIR/$bin"
754                     std_logger "Found previously downloaded binary ($TMPDIR/$bin).␣
↪Removed"  "INFO" $LOG_FILE
755                 fi
756
757                 # download binary set
758                 url="https://www.python.org/ftp/python/$version/$bin"
759                 wget "$url" >/dev/null 2>&1
760
761                 # check if downloaded, break
762                 if [ -f "$TMPDIR/$bin" ]; then
763                     echo "$bin"
```

(continues on next page)

(continued from previous page)

```
                    break
                fi
            done
            # check if downloaded, break
            if [ -f "$TMPDIR/$bin" ]; then
                break
            fi
        done
        return 0
    }

    nminus1="$major.$(( $minor - 1 ))"
    bin_dcr="Python-${nminus1}.tar.xz"

    # for Python < version 2.6, only tgz archive available; download alternate bin format
    if (( $(echo "$major < 2.6" | bc -l) )); then  bin_file=$bin_alt; fi

    arr_versions=( "$bin_file" )           # array of download candidates

    cd $TMPDIR || exit $E_OSERROR

    for bin in "${arr_versions[@]}"; do

        # check for pre-existing file
        if [ -f "$TMPDIR/$bin" ]; then
            cached_download=true
            std_message "Found previously downloaded binary ($bin). Using..." "OK" $LOG_FILE
            break
            #rm "$TMPDIR/$bin"
        fi

        std_message "Downloading $bin ..." "INFO"  $LOG_FILE

        if [ "$(type wget 2>/dev/null)" ]; then

            url="https://www.python.org/ftp/python/$version/$bin"
            wget "$url" >/dev/null 2>&1 &
            progress_dots --fast "false"

        elif [ "$(type curl 2>/dev/null)" ]; then

            curl -O "https://www.python.org/ftp/python/$version/$bin" >/dev/null 2>&1 &
            progress_dots --fast "false"

        else
            std_error_exit "Either wget or curl binary not found. Cannot retrieve python source files" $E_DEPENDENCY
        fi

        # check if downloaded, break
        if [ -f "$TMPDIR/$bin" ]; then
            break
        else
            std_warn "$bin not downloaded, version $version in alpha, beta, or release candidate stage." $LOG_FILE
```

(continues on next page)

(continued from previous page)

```
817             std_message "Downloading latest available alpha, beta, or release␣
↪candidate binary" "INFO"
818             bin=''
819
820             # attempt to download release canidate
821             bin="$(download_rc)"
822
823             if [ ! -f "$TMPDIR/$bin" ]; then
824                 std_warn "Release candidate version not found. Exit" $LOG_FILE
825                 exit $E_OSERROR
826             fi
827         fi
828
829     done
830
831     # user status msg
832     if file_check "$TMPDIR/$bin"; then
833
834         if [[ $(echo $bin | grep "Python-$version.tar.xz") ]] && [[ "$cached_download
↪" ]]; then
835             std_message "${yellow}Python${btext} ${bd}$version${btext} cached source␣
↪successfully located in ${fs}$TMPDIR/$bin${reset}" "INFO" $LOG_FILE
836
837         elif [[ $(echo $bin | grep "Python-$version.tar.xz") ]]; then
838             std_message "${yellow}Python${btext} ${bd}$version${btext} source␣
↪downloaded successfully to ${fs}$TMPDIR/$bin${reset}" "INFO" $LOG_FILE
839
840         elif [[ $(echo $bin | grep "Python-${version}rc[1-4].tar.xz") ]]; then
841             std_message "${yellow}Python${btext} ${bd}$version${btext} release␣
↪candidate source downloaded successfully to ${fs}$TMPDIR/$bin${reset}" "INFO" $LOG_
↪FILE
842
843         elif [[ $(echo $bin | grep "Python-$version[a-b][1-4].tar.xz") ]]; then
844             std_message "${yellow}Python${btext} ${bd}$version${btext} alpha/beta␣
↪source downloaded successfully to ${fs}$TMPDIR/$bin${reset}" "INFO" $LOG_FILE
845         fi
846
847     else
848         # download filed
849         std_warn "Problem downloading ${yellow}Python${btext} ${bd}$version${btext}␣
↪source binary to ${fs}$TMPDIR/$bin${btext}" $LOG_FILE
850         std_warn "${fs}$TMPDIR/$bin${btext} Not saved to local filesystem" $LOG_FILE
851     fi
852     #
853     # <<-- end function download_binary -->>
854 }
855
856
857 function file_check(){
858     ##
859     ##  Validates existence of file object on local fs
860     ##
861     ##  Returns:
862     ##      - Success | Failure, TYPE: bool
863     ##
864
865     local file_path="$1"
```

(continues on next page)

(continued from previous page)

```
    if [ -f "$file_path" ]; then

        return 0

    else

        return 1      # file not found

    fi
    #
    # <<-- end function file_check -->>
}


function pip_backup(){
    ##
    ## create backup of local python package repo ##
    ##
    local pip                       # pip package mgr binary path
    local pip3                      # pip3 package mgr binary path
    local pip_r
    local pip3_backup

    pip3=$(which pip3 2>/dev/null)
    pip3_backup="pip3-requirements-$NOW.txt"

    pip=$(which pip 2>/dev/null)
    pip_r="pip-requirements-$NOW.txt"
    #
    msg="Creating backups of local python package repositories in $workdir"
    std_message "$msg" "INFO"
    std_logger "Backup of local python package repositories - START" "INFO" $LOG_FILE

    # pip backup
    if [ "$pip" ]; then
        $pip --no-cache-dir freeze > $workdir/$pip_r
        std_logger "Created pip backup: $workdir/$pip_r" "INFO" $LOG_FILE
    fi

    # pip3 backup
    if [ "$pip3" ]; then
        $pip3 --no-cache-dir freeze > $workdir/$pip3_backup
        std_logger "Created pip3 backup: $workdir/$pip3_backup" "INFO" $LOG_FILE
    fi

    if [ -f "$workdir/$pip3_backup" ] && [ -f "$workdir/$pip_r" ]; then

        if [[ $(diff $workdir/$pip_r $workdir/$pip3_backup) ]]; then
            # pip backup and pip3 files are different
            msg="Created backups of local python package repositories in $workdir:\n
            \tPip: \t$pip_r \n\t\tPip3: \t$pip3_backup\n"
            std_message "$msg" "INFO" $LOG_FILE

        else
            # pip backup & pip3 backup file same
            rm -f $workdir/$pip3_backup
```

(continues on next page)

(continued from previous page)

```
            std_logger "Deleting pip3 backup file ($pip3_backup) - Identical to $pip_r
↪" "INFO" $LOG_FILE
            std_logger "Retained single backup file $workdir/$pip_r" "INFO" $LOG_FILE

            msg="Created backup of python package repositories in
↪$workdir:\n\n\t\tpip: \t$pip_r"
            std_message "$msg" "INFO"
        fi

    elif [ $pip3_backup ]; then
        # pip3 backup only
        msg="Created backup of python package repositories in $workdir:\n\n\t\tpip: \t
↪$pip3_backup"
        std_message "$msg" "INFO"
        std_logger "Created backup file in $workdir:  $pip3_backup" "INFO" $LOG_FILE

    else
        # pip backup only
        msg="Created backup of python package repositories in $workdir:\n\n\t\tpip: \t
↪$pip_r"
        std_message "$msg" "INFO"
        std_logger "Created backup file in $workdir:  $pip_r" "INFO" $LOG_FILE
    fi

    std_logger "Backup of local python package repositories - END" "INFO" $LOG_FILE
    return 0

    #
    # <<-- end function pip_backup -->>
}


function latest_version_available(){
    ##
    ## retrieve latest minor version of python to install ##
    ##
    local major_version="$1"                # 3.7, etc
    local minor=20                          # starting count to decrement 3.7.minor
    local act="$(echo -e ${cyan})"          # color code
    local rst="$(echo -e ${resetansi})"     # reset color code
    #
    # Guarding Clause:
    # Return if VERSION during installs already set; else continue
    if [ "$VERSION" ] && [ "$INSTALL" ]; then return 0; fi

    if [ "$(type wget 2>/dev/null)" ]; then
        wget -O "$TMPDIR/index.html" 'https://www.python.org/ftp/python' 2>/dev/null
    else
        curl -L -o "$TMPDIR/index.html" 'https://www.python.org/ftp/python' 2>/dev/
↪null
    fi

    # find highest (most recent) minor version
    while (( $minor >= 0 )); do

        try="$(grep $major_version.$minor $TMPDIR/index.html)"

```

(continues on next page)

(continued from previous page)

```
        if [ "$try" ]; then
            # set full version identifer to global var
            VERSION=$major_version.$minor

            # date of release
            rdate="$(echo $try | awk '{print $3}')"
            fdate="${act}$(date --date=${rdate} +"%a %b %d %Y")${rst}"

            std_message "Found most recent binaries available for Python Version ${major_version}" "OK" $LOG_FILE
            std_message "${yellow}Python ${title}$VERSION${rst}, released on $fdate" "INFO" $LOG_FILE

            found="true"
            break
        fi
        minor=$(( $minor - 1 ))
    done

    if [ ! "$found" ]; then
        std_warn "No binary version found for Python $major_version. Exit" $LOG_FILE
        rm $TMPDIR/index.html
        return 1    # minor version not found; return false
    fi

    # rm version artifacts
    rm $TMPDIR/index.html
    return 0        # minor version identified; return true
}


function logfile_owner(){
    ##
    ## ensures log file owned by normal user ##
    ##
    local log_file="$1"
    local console_log="$2"
    local yl=${yellow}
    local rst=${reset}
    #
    # update owner of log file if NOT located in /var/log
    if [ "$SUDO_USER" ] && [ ! "$(echo $log_file | grep \/var)" ] && \
       [ "$(ls -l $log_file | awk '{print $3}')" != "root" ]; then

          chown $SUDO_USER:$SUDO_USER $log_file
          std_logger "Successfully updated file owner to root for ${yl}$log_file${rst}" "INFO" $log_file

    fi
    # update owner of console log file if NOT located in /var/log
    if [ "$SUDO_USER" ] && [ ! "$(echo $console_log | grep \/var)" ] && \
       [ "$(ls -l $console_log | awk '{print $3}')" != "root" ]; then

            chown $SUDO_USER:$SUDO_USER $console_log
            std_logger "Successfully updated file owner to root for ${yl}$console_log${rst}" "INFO" $log_file

```

(continues on next page)

(continued from previous page)

```
    fi
    # update permissions on log files if not correct
    if [ "$SUDO_USER" ] || [ "$EUID" = "0" ]; then
        chmod 0666 $log_file $console_log

    fi
}


function mlocate_installation(){
    ##
    ## installs os-specfic installation of mlocate; runs db update
    ##
    ##   This should ideally work via the following:
    ##      1. install mlocate
    ##      2. after updatedb -v run, find all python3.X locations
    ##      3. Using this list, delete each path in the list
    ##
    ##  Installing mlocate is OS-specific
    ##
    local clearscn=$(which clear 2>/dev/null)
    local epel_url='https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.
↪noarch.rpm'
    #
    os_type     # linux distro environment; sets globals: family, release
    #
    std_message "$family Linux:  Installing mlocate binaries in preparation for␣
↪system Python3 removal" "INFO" $LOG_FILE

    if [ "$family" = "AmazonLinux" ]; then

        $clearscn    # clear screen
        std_message "$family Linux:  Update OS pkgs" "INFO" $LOG_FILE
        yum -y update >> $LOG_CONSOLE &
        progress_dots --fast "false"
        std_message "$family Linux:  Install mlocate and dependencies" "INFO" $LOG_
↪FILE
        yum -y install mlocate
        std_message "$family Linux:  First db filesystem discovery run..." "INFO"
↪$LOG_FILE
        updatedb -v >/dev/null &
        progress_dots --fast "false" --text "Searching for Python filesystem locations
↪"
        return 0

    elif [ "$family" = "Redhat" ] || [ "$family" = "CentOS" ]; then

        $clearscn    # clear screen
        std_message "$family Linux:  Update OS pkgs" "INFO" $LOG_FILE
        yum -y update >> $LOG_CONSOLE &
        progress_dots --fast "false"
        # install epel
        std_message "$family Linux:  Installing epel package repository" "INFO" $LOG_
↪FILE
        curl -o epel.rpm $epel_url
        yum -y install epel.rpm
        std_message "$family Linux:  Install mlocate and dependencies" "INFO" $LOG_
↪FILE
```

(continues on next page)

(continued from previous page)

```
        yum -y update
        yum -y install mlocate
        std_message "$family Linux:  First db filesystem discovery run..." "INFO"
↪$LOG_FILE
        updatedb -v >/dev/null &
        progress_dots --fast "false" --text "Searching for Python filesystem locations
↪"
        return 0

    elif [ "$family" = "Ubuntu" ] || [ "$family" = "Mint" ]; then

        $clearscn    # clear screen
        std_message "$family Linux:  Update OS pkgs" "INFO" $LOG_FILE
        apt update -y >> $LOG_CONSOLE &
        progress_dots --fast "false"
        std_message "$family Linux:  Install mlocate and dependencies" "INFO" $LOG_
↪FILE
        apt install -y mlocate
        std_message "$family Linux:  First db filesystem discovery run..." "INFO"
↪$LOG_FILE
        updatedb -v >/dev/null &
        progress_dots --fast "false" --text "Searching for Python filesystem locations
↪"
        return 0

    fi

    return 1    # mlocate installation fail or not possible
    #
    # <-- end function mlocate_installation -->
}


function os_type(){
    ##
    ## determine linux os distribution ##
    ##
    function determine_osrelease(){
        # interrogate /etc/os-release directly
        # to determine release
        id=$(grep VERSION_ID /etc/os-release | awk -F '=' '{print $2}')
        clean_id=$(echo $id | cut -c 2-20 | rev | cut -c 2-20 | rev)
        printf -- '%s\n' "$clean_id"
    }

    OS_INFO=$(source $pkg_lib/os_distro.sh 2>/dev/null)


    if [ "$OS_INFO" ]; then

        family=$(echo $OS_INFO | awk '{print $1}')
        release=$(echo $OS_INFO | awk '{print $2}')
        codename=$(echo $OS_INFO | awk '{print $3}')

        case "$family" in
            'Ubuntu')
                act=${a_magenta}
```

(continues on next page)

(continued from previous page)

```
                ;;
            'Redhat' | 'CentOS')
                act=${red}
                ;;
            'AmazonLinux')
                act=${a_orange}
                ;;
        esac

        if [ ! "$release" ] || [ -z "$release" ]; then
            release=$(determine_osrelease)
        fi

        std_message "Operating System Info detected:
        \n\t\tFamily:\t\t${act}$family${btext}
        \tRelease:\t${bd}$release${btext}
        \tCodename:\t$codename" "INFO"
        std_logger "OS Info Detected: Family: $family, Release: $release, Codename:
↪$codename" "INFO" $LOG_FILE

    else

        std_warn "Failed to determine Linux Distribution Type.  Fall back detection"
↪$LOG_FILE
        FALLBACK_DETECT="true"

    fi
    #
    # <-- end function os_type -->
}


function ospackage_installed(){
    ##
    ##  os specific test to determine if an operating system
    ##  package is installed
    ##
    local packagemgr="$1"       # packagemgr; redhat (.rpm) or debian (.deb)
    local ospkg="$2"            # os package tested for installation status

    case $packagemgr in
        'rpm' | '.rpm' | 'redhat')
            if [ ! "$(rpm -q $ospkg | grep 'not installed')" ]; then
                return 0    # is installed
            fi
            ;;
        'deb' | '.deb' | 'debian')
            if [ "$(dpkg -s $ospkg 2>/dev/null | grep Status\:)" ]; then
                return 0    # is installed
            fi
            ;;
    esac

    return 1    # not installed
    #
    # < --- end function ospackage_installed --->
}
```

(continues on next page)

(continued from previous page)

```
1186
1187
1188 function ospackage_available(){
1189     ##
1190     ##  os specific test to determine if an operating system
1191     ##  package is present in local package repositories
1192     ##
1193     local packagemgr="$1"       # packagemgr; redhat (.rpm) or debian (.deb)
1194     local ospkg="$2"            # os package tested for installation status
1195
1196     case $packagemgr in
1197         'rpm' | '.rpm' | 'redhat')
1198             if [ "$(yum list $ospkg 2>/dev/null | grep -i 'Available')" ]; then
1199                 return 0    # is available, but not installed
1200             fi
1201             ;;
1202         'deb' | '.deb' | 'debian')
1203             if [ "$(apt-cache search $ospkg)" ]; then
1204                 return 0    # is available, but not installed
1205             fi
1206             ;;
1207     esac
1208
1209     # not installed
1210     return 1
1211     #
1212     # < --- end function ospackage_available --->
1213 }
1214
1215
1216 function ospackage_status(){
1217     ##
1218     ##  displays os package summary info
1219     ##
1220     local packagemgr="$1"
1221     local ospkg="$2"
1222     local logfile="$3"
1223     local prefix
1224
1225     case $packagemgr in
1226
1227         'deb')
1228             if ospackage_installed $packagemgr $ospkg; then
1229                 # available and installed
1230                 summary=$(apt-cache show $ospkg | grep Description | head -n 1 | awk -
↪F ':' '{print $2}')
1231                 prefix=" INSTALLED"
1232
1233             elif ospackage_available $packagemgr $ospkg; then
1234                 # available but not installed
1235                 if [ "$(apt-cache show $ospkg 2>/dev/null)" ]; then
1236                     summary=$(apt-cache show $ospkg | grep Description | head -n 1 |␣
↪awk -F ':' '{print $2}')
1237                 else
1238                     summary=$(apt-cache search $ospkg 2>/dev/null | awk '{ print␣
↪substr(" "$0, index($0,$3)) }')
1239                 fi
```

(continues on next page)

(continued from previous page)

```
                prefix=" AVAILABLE"

            elif ! ospackage_available $packagemgr $ospkg; then
                # not available
                prefix=" NOT-FOUND"
            fi
            ;;

        'rpm')
            if ospackage_installed $packagemgr $ospkg; then
                # available and installed
                summary=$(rpm -qi $ospkg | grep 'Summary' | awk -F ':' '{print $2}')
                prefix=" INSTALLED"

            elif ospackage_available $packagemgr $ospkg; then
                # available but not installed
                summary=$(yum info $ospkg 2>/dev/null | grep 'Summary' | head -n1 |␣
↪awk -F ':' '{print $2}')
                prefix=" AVAILABLE"

            elif ! ospackage_available $packagemgr $ospkg; then
                # not available
                prefix=" NOT-FOUND"
            fi
            ;;

        'alternate_rpm')
            if ospackage_available 'rpm' $ospkg; then
                # available but not installed
                summary=$(yum info $ospkg 2>/dev/null | grep 'Summary' | head -n1 |␣
↪awk -F ':' '{print $2}')
            else
                summary='no available package description in current os'
            fi
            prefix="ESTIMATE"
            ;;

        'alternate_deb')
            if ospackage_available 'deb' $ospkg; then
                # available but not installed
                if [ "$(apt-cache show $ospkg 2>/dev/null)" ]; then
                    summary=$(apt-cache show $ospkg | grep Description | head -n 1 |␣
↪awk -F ':' '{print $2}')
                else
                    summary=$(apt-cache search $ospkg 2>/dev/null | awk '{ print␣
↪substr(" "$0, index($0,$3)) }')
                fi
            else
                summary='no available package description in current os'
            fi
            prefix="ESTIMATE"
            ;;

        'alternate')
            printf -- ' %s\n' "$ospkg"
            ;;
    esac
```

(continues on next page)

(continued from previous page)

```
1293
1294        if [ -z "$summary" ] || [ "$summary" = "" ]; then
1295            summary=' description not found'
1296        fi
1297
1298        case $summary in
1299            ' description not found')
1300                std_logger "$ospkg: Description not found for os-package $ospkg" "$prefix
    ↪" $logfile
1301                ;;
1302
1303            *)
1304                case ${#ospkg} in
1305                    [0-5])
1306                        std_message "$ospkg\t\t\t|  $summary" $prefix $logfile
    ↪'noblanklines'
1307                        ;;
1308                    [6-9] | 10 | 11 | 12 | 13)
1309                        std_message "$ospkg\t\t|  $summary" $prefix $logfile 'noblanklines
    ↪'
1310                        ;;
1311                    *)
1312                        std_message "$ospkg\t|  $summary" $prefix $logfile 'noblanklines'
1313                        ;;
1314                esac
1315                ;;
1316        esac
1317        #
1318        # <--- end function ospackage_status --- >
1319    }
1320
1321
1322    function install_deb_packages(){
1323        ##
1324        ##  installs packages supplied as an array
1325        ##
1326        ##  Usage:
1327        ##
1328        ##      install_deb_packages  -o "install"  --packages pkgs[@]
1329        ##
1330        ##  Where:
1331        ##      --operation:  update | upgrade | install
1332        ##      --packages:  associative array
1333        ##
1334        ##  Note:
1335        ##      Mint Linux machines not upgraded via apt for reliability
1336        ##
1337        declare -a pkg_list
1338        local operation
1339        local logfile=$LOG_FILE
1340        local consolelog=$LOG_CONSOLE
1341        local show_packages=$SHOW_OSPACKAGES
1342
1343        while [ $# -gt 0 ]; do
1344            case $1 in
1345                -a | --alt_os)
1346                    alternate_os="$2"
```

(continues on next page)

(continued from previous page)

```
                shift 2
                ;;

            -l | --log)
                logfile="$2"; shift 2
                ;;

            -o | --operation)
                case $2 in
                    update | upgrade | install)
                        operation="$2"
                        shift 2
                        ;;
                    *)
                        std_error_exit "Operation must be either update, upgrade, or install. Exit"
                        ;;
                esac
                ;;

            -p | --packages)
                pkg_list=("${!2}"); shift 2
                ;;
        esac
    done

    ##  display ospackage status instead of updating   ##

    if [ $show_packages ] && [ $alternate_os ]; then
        case $alternate_os in
            redhat* | centos* | amazonlinux*)
                ospackage_status "alternate_rpm" "$ospkg" "$logfile"
                ;;
            *)
                ospackage_status "alternate_deb" "$ospkg" "$logfile"
                ;;
        esac

    elif [ $show_packages ]; then
        if [ -z $operation ] || [ "$operation" = "install" ]; then
            for ospkg in "${pkg_list[@]}"; do

                ospackage_status "deb" "$ospkg" "$logfile"

            done
        fi
        return 0
    fi

    ##    perform actual update of system ospackages   ##

    if [ "$operation" = "update" ]; then
        std_message "Refreshing os package repository caches" "INFO" $logfile
        apt update -y 2>/dev/null | tee $tty >> $consolelog
        std_message "Package repository cache refresh complete" "OK" $logfile
        return 0

```

(continues on next page)

(continued from previous page)

```
    elif [ "$operation" = "upgrade" ] && [ "$family" = "Ubuntu" ]; then
        std_message "Upgrading Operating System Packages" "INFO" $logfile
        apt upgrade -y 2>/dev/null  | tee $tty >> $consolelog
        std_message "Operating System update complete" "OK" $logfile
        return 0

    elif [ -z $operation ] || [ "$operation" = "install" ]; then
        for ospkg in "${pkg_list[@]}"; do

            # validate install
            if ospackage_installed 'deb' $ospkg; then
                std_message "Package $ospkg already installed" "OK" $logfile
            else
                std_message "Installing Debian package $ospkg" "INFO" $logfile
                apt install -y $ospkg  2>/dev/null | tee $tty >> $consolelog

                # verify install
                if ospackage_installed 'deb' $ospkg; then
                    std_message "Package $ospkg installed successfully" "OK" $logfile
                else
                    std_warn "Package $ospkg failed to install. Possibly missing from os package repository" $logfile
                fi
            fi

        done
        return 0
    fi

    return 1
    #
    # <<--- end function install_deb_packages --->>
}


function install_yum_packages(){
    ##
    ##  installs packages supplied as an array
    ##
    ##  Usage:
    ##
    ##      install_yum_packages  -o "install"  --packages pkgs[@]
    ##
    ##  Where:
    ##      --operation:  update | install
    ##      --packages:  associative array
    ##
    declare -a pkg_list
    local operation
    local pkgmanager
    local logfile=$LOG_FILE
    local consolelog=$LOG_CONSOLE
    local show_packages=$SHOW_OSPACKAGES

    if [ "$(which dnf 2>/dev/null)" ]; then
        pkgmanager=$(which dnf 2>/dev/null)

```

(continues on next page)

(continued from previous page)

```
    elif [ "$(which yum 2>/dev/null)" ]; then
        pkgmanager=$(which yum 2>/dev/null)

    else
        std_warn "No package manager found. Abort os package install" $logfile
    fi

    while [ $# -gt 0 ]; do
        case $1 in
            -a | --alt_os)
                alternate_os="$2"
                shift 2
                ;;

            -l | --log)
                logfile="$2"
                shift 2
                ;;

            -o | --operation)
                case $2 in
                    update | install | groupinstall)
                        operation="$2"
                        shift 2
                        ;;
                    *)
                        std_error_exit "Operation must be either update, install, or␣
↪groupinstall. Exit"
                        ;;
                esac
                ;;

            -p | --packages)
                pkg_list=("${!2}")
                shift 2
                ;;
        esac
    done

    ##  display ospackage status instead of updating   ##

    if [ $show_packages ] && [ $alternate_os ]; then
        ospackage_status "alternate" "$ospkg" "$logfile"

    elif [ $show_packages ]; then
        if [ -z $operation ] || [ "$operation" = "install" ] || [ "$operation" =
↪"groupinstall" ]; then

            for ospkg in "${pkg_list[@]}"; do
                ospackage_status "rpm" "$ospkg" "$logfile"
            done

        fi
        return 0
    fi

    ##    perform actual update of system ospackages   ##
```

(continues on next page)

(continued from previous page)

```
1514
1515        if [ "$operation" = "update" ]; then
1516            std_message "Refreshing os package repository caches" "INFO" $logfile
1517            $pkgmanager -y update 2>/dev/null | tee $tty >> $consolelog
1518            std_message "Package repository cache refresh complete" "OK" $logfile
1519            return 0
1520
1521        elif [ "$operation" = "groupinstall" ]; then
1522            for osgroup in "${pkg_list[@]}"; do
1523                std_message "Installing operating system package group" "INFO" $logfile
1524                $pkgmanager -y groupinstall $osgroup 2>/dev/null  | tee $tty >>
↪$consolelog
1525            done
1526            std_message "Operating System Upgrade Complete" "OK" $logfile
1527            return 0
1528
1529        elif [ -z $operation ] || [ "$operation" = "install" ]; then
1530            for ospkg in "${pkg_list[@]}"; do
1531
1532                # validate install
1533                if ospackage_installed 'rpm' $ospkg; then
1534                    std_message "Package $ospkg already installed - nothing to do" "INFO"
↪$logfile
1535                else
1536                    std_message "Installing package $ospkg" "INFO" $logfile
1537                    $pkgmanager -y install $ospkg  2>/dev/null | tee $tty >> $consolelog
1538
1539                    # verify install
1540                    if ospackage_installed 'rpm' $ospkg; then
1541                        std_message "Package $ospkg installed successfully" "OK" $logfile
1542                    else
1543                        std_warn "Package $ospkg failed to install. Possibly missing from␣
↪os package repository" $logfile
1544                    fi
1545                fi
1546
1547            done
1548            return 0
1549        fi
1550
1551        return 1
1552        #
1553        # <<--- end function install_yum_packages --->>
1554    }
1555
1556
1557    function print_header(){
1558        ##
1559        ## print formatted report header ##
1560        ##
1561        local width="$1"
1562        local color="$2"
1563        #
1564        echo -e "${color}"
1565        printf '%*s' "$width" '' | tr ' ' -  | indent04
1566        echo -e "${btext}"
1567        printf -- '      %s   %s\t%s\t\t%s\t\t\t%s' "${BOLD}Status" '|' "Package" '|'
↪"Description${UNBOLD}"
```

(continues on next page)

(continued from previous page)

```
    echo -e "${color}"
    printf '%*s' "$width" '' | tr ' ' - | indent04
    echo -e "${btext}"
    #
    # <<--- end function print_header -->>
}


function os_packages(){
    ##
    ## install os prerequisites depending on os type ##
    ##
    local version
    declare -a packages

    version="$1"

    if [ $SHOW_OSPACKAGES ]; then
        if [ ! $family ]; then std_error_exit "Unknown operating" $E_DEPENDENCY; fi
        # only show which os packages to be installed
        print_header "100" "${a_wgray}"
    else
        std_message "Installing OS-specific packages for Python-$version..." "INFO"
↪$LOG_FILE
    fi

    # AMAZON Linux, Version 1

        if [ "$family" = "AmazonLinux" ] && [ "$release" = "1" ]; then
            install_yum_packages --operation update

            packages=( "Development Libraries" "Development tools" )
            install_yum_packages --operation groupinstall --packages packages[@]

            packages=( "gcc" "make" "wget" "which" )
            install_yum_packages --operation install --packages packages[@]

            packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'ncurses' 'ncurses-devel' 'ncurses-libs' 'ncurses-static')
            install_yum_packages --operation install --packages packages[@]

            packages=( 'ncurses-libs' 'ncurses-term' 'uuid-devel' )
            install_yum_packages --operation install --packages packages[@]

            packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'zlib' 'zlib-devel' 'zlib-static' 'libffi-devel')
            install_yum_packages --operation install --packages packages[@]
            return 0

    # AMAZON Linux, Version 2

        elif [ "$family" = "AmazonLinux" ] && [ "$release" = "2" ]; then
            install_yum_packages --operation update
```

(continues on next page)

(continued from previous page)

```
            packages=( "Development tools" )
            install_yum_packages --operation groupinstall --packages packages[@]

            packages=( "gcc" "make" "wget" "which" )
            install_yum_packages --operation install --packages packages[@]

            packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
            install_yum_packages --operation install --packages packages[@]

            packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'ncurses' 'ncurses-devel' 'ncurses-libs' 'ncurses-static')
            install_yum_packages --operation install --packages packages[@]

            packages=( 'ncurses-libs' 'ncurses-term' 'uuid-devel' )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'python-magic' 'libuuid-devel' 'tk-devel' )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'xz' 'xz-devel' 'xz-libs')
            install_yum_packages --operation install --packages packages[@]

            packages=( 'zlib' 'zlib-devel' 'libffi-devel')
            install_yum_packages --operation install --packages packages[@]

            packages=( 'lzma-sdk' 'lzma-sdk-devel' 'xz-devel' 'gdbm-devel' )
            install_yum_packages --operation install --packages packages[@]
            return 0

        elif [ $FALLBACK_DETECT ] && [ "$(grep -i amazon /etc/os-release  | head -n 1)
↪" ]; then
            install_yum_packages --operation update

            packages=( "Development Libraries" "Development tools" )
            install_yum_packages --operation groupinstall --packages packages[@]

            packages=( "gcc" "make" "wget" "which" 'uuid-devel' )
            install_yum_packages --operation install --packages packages[@]

            packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
            install_yum_packages --operation install --packages packages[@]

            packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
            install_yum_packages --operation install --packages packages[@]
            return 0

    # REDHAT Enterprise Linux 7

        elif [ "$family" = "Redhat" ] && { [ "$release" = "7.3" ] || [ "$release" =
↪"7.4" ]; }; then
            install_yum_packages --operation update

            packages=( "Development tools" )
            install_yum_packages --operation groupinstall --packages packages[@]
```

(continues on next page)

(continued from previous page)

```
            packages=( "gcc" "make" "wget" "which" )
            install_yum_packages --operation install --packages packages[@]

            packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'ncurses' 'ncurses-devel' 'ncurses-libs' 'ncurses-static')
            install_yum_packages --operation install --packages packages[@]

            packages=( 'ncurses-libs' 'ncurses-term' 'uuid-devel' )
            install_yum_packages --operation install --packages packages[@]

            packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'xz' 'xz-devel' 'xz-libs' 'libuuid-devel' 'tk-devel' )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'zlib' 'zlib-devel' 'libffi-devel')
            install_yum_packages --operation install --packages packages[@]
            return 0

        elif [ "$family" = "Redhat" ] && { [ "$release" = "7.5" ] || [ "$release" =
↪"7.6" ]; }; then
            install_yum_packages --operation update

            packages=( "Development tools" )
            install_yum_packages --operation groupinstall --packages packages[@]

            packages=( "gcc" "make" "wget" "which" )
            install_yum_packages --operation install --packages packages[@]

            packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'ncurses' 'ncurses-devel' 'ncurses-libs' 'ncurses-static')
            install_yum_packages --operation install --packages packages[@]

            packages=( 'ncurses-libs' 'ncurses-term' 'uuid-devel' )
            install_yum_packages --operation install --packages packages[@]

            packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'xz' 'xz-devel' 'xz-libs' 'libuuid-devel' 'tk-devel' )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'zlib' 'zlib-devel' 'libffi-devel')
            install_yum_packages --operation install --packages packages[@]

            packages=( 'lzma-sdk' 'lzma-sdk-devel' 'xz-devel' 'gdbm-devel' )
            install_yum_packages --operation install --packages packages[@]
            return 0

    # CentOS Linux (Redhat binary compatible)
```

(continues on next page)

(continued from previous page)

```
        elif [ "$family" = "CentOS" ] && { [ "$release" = "7" ] || [ "$release" = "8" ]; }; then
            install_yum_packages --operation update

            packages=( "Development tools" )
            install_yum_packages --operation groupinstall --packages packages[@]

            packages=( "gcc" "make" "wget" )
            install_yum_packages --operation install --packages packages[@]

            packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
            install_yum_packages --operation install --packages packages[@]

            packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'ncurses' 'ncurses-devel' 'ncurses-libs' )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'ncurses-base' 'ncurses-term')
            install_yum_packages --operation install --packages packages[@]

            packages=( 'xz' 'xz-devel' 'xz-libs')
            install_yum_packages --operation install --packages packages[@]

            packages=( 'zlib' 'zlib-devel' 'libffi-devel')
            install_yum_packages --operation install --packages packages[@]

            packages=( 'xz-devel' 'gdbm-devel' 'libuuid-devel' 'tk-devel' )
            install_yum_packages --operation install --packages packages[@]

            if [ "$release" = "7" ]; then
                packages=( 'ncurses-static' 'yum-utils' 'python36-pyOpenSSL' )
                install_yum_packages --operation install --packages packages[@]

                packages=( 'lzma-sdk' 'lzma-sdk-devel' 'uuid-devel' )
                install_yum_packages --operation install --packages packages[@]
            else
                packages=( 'python3-pyOpenSSL' 'libuuid-devel' )
                install_yum_packages --operation install --packages packages[@]
            fi
            return 0

    # Fedora Linux

        elif [ "$family" = "Fedora" ] && (( "$release" >= "26" )); then
            install_yum_packages --operation update

            packages=( "Development tools" )
            install_yum_packages --operation groupinstall --packages packages[@]

            packages=( "yum-utils" "gcc" "make" "wget" )
            install_yum_packages --operation install --packages packages[@]

            packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
            install_yum_packages --operation install --packages packages[@]

```

(continues on next page)

(continued from previous page)

```
            packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'ncurses' 'ncurses-devel' 'ncurses-libs' 'ncurses-static')
            install_yum_packages --operation install --packages packages[@]

            packages=( 'ncurses-libs' 'ncurses-term' 'uuid-devel' )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'python3-magic' 'libuuid-devel' 'tk-devel' )
            install_yum_packages --operation install --packages packages[@]

            packages=( 'xz' 'xz-devel' 'xz-libs')
            install_yum_packages --operation install --packages packages[@]

            packages=( 'zlib' 'zlib-devel' 'libffi-devel')
            install_yum_packages --operation install --packages packages[@]

            packages=( 'lzma-sdk' 'lzma-sdk-devel' 'xz-devel' 'gdbm-devel' )
            install_yum_packages --operation install --packages packages[@]
            return 0

    # UBUNTU, Ubuntu variants, Linux Mint

        elif { [ "$family" = "Ubuntu" ] && [ "$(echo "$release" | grep 14)" ]; } || \
             { [ "$family" = "Mint" ] && [ "$(echo "$release" | grep 17)" ]; }; then

            install_deb_packages --operation update
            install_deb_packages --operation upgrade

            packages=( "build-essential" "checkinstall" )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'gcc' 'make' 'wget' 'python3-openssl' )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'libreadline-dev' 'libreadline6-dev' 'libssl-dev' 'libffi-dev'
↪)
            install_deb_packages --operation install --packages packages[@]

            packages=( 'libbz2-dev' 'libgdbm-dev' 'zlib1g-dev' 'uuid-dev' )
            install_deb_packages --operation install --packages packages[@]

            packages=( "libncursesw5-dev" "tk-dev" "libc6-dev" "libbz2-dev" )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'sqlite3' 'sqlite3-doc' 'libsqlite3-dev' )
            install_deb_packages --operation install --packages packages[@]
            return 0

        elif { [ "$family" = "Ubuntu" ] && [ "$(echo "$release" | grep '16')" ]; } ||
↪\
             { [ "$family" = "Mint" ] && [ "$(echo "$release" | grep '18')" ]; }; then

            install_deb_packages --operation update
            install_deb_packages --operation upgrade

```

(continues on next page)

(continued from previous page)

```
            packages=( 'build-essential' 'checkinstall' 'llvm')
            install_deb_packages --operation install --packages packages[@]

            packages=( 'gcc' 'make' 'wget' 'python3-openssl' )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'zlib1g-dev' 'libssl-dev' 'libffi-dev')
            install_deb_packages --operation install --packages packages[@]

            packages=( 'libbz2-dev' 'libgdbm-dev' 'libsqlite3-dev' 'libc6-dev' )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'libncursesw5-dev' 'libncurses5-dev' 'lib64ncurses5'
↪'lib64ncurses5-dev' )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'libreadline6' 'libreadline6-dev' 'libreadline-dev' )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'ncurses-term' 'ncurses-examples' 'ncurses-base' 'ncurses-doc')
            install_deb_packages --operation install --packages packages[@]

            packages=( 'readline-doc' 'gdb' 'gdb-doc' 'uuid-dev' )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'sqlite3' 'sqlite3-doc' 'libsqlite3-dev' )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'tk-dev' 'python3-tk' 'python3-magic' 'xz-utils' )
            install_deb_packages --operation install --packages packages[@]

            if [ $EXTRA_PACKAGES ]; then
                packages=(
                    binutils-doc gettext cpp-doc gcc-5-locales debian-keyring
                    g++-multilib g++-5-multilib gcc-5-doc libstdc++6-5-dbg gcc-
↪multilib
                    autoconf automake libtool flex bison gcc-doc gcc-5-multilib
                    libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg libasan2-dbg
                    liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrts5-dbg
                    libmpx0-dbg libquadmath0-dbg glibc-doc libstdc++-5-doc make-doc
                )
                install_deb_packages --operation install --packages packages[@]
            fi
            return 0

        elif { [ "$family" = "Ubuntu" ] && [ "$(echo "$release" | grep '18')" ]; } ||␣
↪\
             { [ "$family" = "Mint" ] && [ "$(echo "$release" | grep '19')" ]; }; then

            install_deb_packages --operation update
            install_deb_packages --operation upgrade

            packages=( 'build-essential' 'checkinstall' 'llvm')
            install_deb_packages --operation install --packages packages[@]

            packages=( 'gcc' 'make' 'wget' )
            install_deb_packages --operation install --packages packages[@]
```

(continues on next page)

(continued from previous page)

```
1900
1901            packages=( 'zlib1g-dev' 'libreadline-dev' 'libssl-dev' 'libffi-dev')
1902            install_deb_packages --operation install --packages packages[@]
1903
1904            packages=( 'libbz2-dev' 'libgdbm-dev' 'libsqlite3-dev' 'libc6-dev' )
1905            install_deb_packages --operation install --packages packages[@]
1906
1907            packages=( 'libreadline6' 'libreadline6-dev' 'libreadline-dev' )
1908            install_deb_packages --operation install --packages packages[@]
1909
1910            packages=( 'libreadline7' 'python3-urwid' 'python3-btrees' 'python3-
↪openssl' )
1911            install_deb_packages --operation install --packages packages[@]
1912
1913            packages=( 'ncurses-term' 'ncurses-examples' 'ncurses-base' 'ncurses-doc')
1914            install_deb_packages --operation install --packages packages[@]
1915
1916            packages=( 'readline-doc' 'gdb' 'gdb-doc' 'uuid-dev' )
1917            install_deb_packages --operation install --packages packages[@]
1918
1919            packages=( 'sqlite3' 'sqlite3-doc' 'libsqlite3-dev' )
1920            install_deb_packages --operation install --packages packages[@]
1921
1922            packages=( 'tk-dev' 'python3-tk' 'python3-magic' 'xz-utils' )
1923            install_deb_packages --operation install --packages packages[@]
1924
1925            packages=( 'liblzma-dev' 'liblz-dev' 'lzma-dev' 'libqdbm-dev' )
1926            install_deb_packages --operation install --packages packages[@]
1927
1928            if [ "$(echo "$release" | grep '18.10')" ]; then
1929                packages=( 'libncurses6' 'libncurses6-dbg' 'libncursesw6'
↪'libncursesw6-dbg' )
1930                install_deb_packages --operation install --packages packages[@]
1931
1932            elif [ "$(echo "$release" | grep '18.04')" ]; then
1933                packages=( 'libncursesw5-dev' 'libncurses5-dev' )
1934                install_deb_packages --operation install --packages packages[@]
1935            fi
1936            return 0
1937
1938        elif { [ "$family" = "Ubuntu" ] && [ "$(echo "$release" | grep '20')" ]; } ||␣
↪\
1939             { [ "$family" = "Mint" ] && [ "$(echo "$release" | grep '20')" ]; }; then
1940
1941            install_deb_packages --operation update
1942            install_deb_packages --operation upgrade
1943
1944            packages=( 'build-essential' 'checkinstall' 'llvm')
1945            install_deb_packages --operation install --packages packages[@]
1946
1947            packages=( 'gcc' 'make' 'wget' )
1948            install_deb_packages --operation install --packages packages[@]
1949
1950            packages=( 'zlib1g-dev' 'libreadline-dev' 'libssl-dev' 'libffi-dev')
1951            install_deb_packages --operation install --packages packages[@]
1952
1953            packages=( 'libbz2-dev' 'libgdbm-dev' 'libsqlite3-dev' 'libc6-dev' )
```

(continues on next page)

(continued from previous page)

```
            install_deb_packages --operation install --packages packages[@]

            packages=( 'libreadline8' 'libreadline-dev' )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'python3-urwid' 'python3-btrees' 'python3-openssl' )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'ncurses-term' 'ncurses-examples' 'ncurses-base' 'ncurses-doc')
            install_deb_packages --operation install --packages packages[@]

            packages=( 'readline-doc' 'gdb' 'gdb-doc' 'uuid-dev' )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'sqlite3' 'sqlite3-doc' 'libsqlite3-dev' )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'tk-dev' 'python3-tk' 'python3-magic' 'xz-utils' )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'libpython3.8' 'libpython3.8-dev' 'libpython3.8-minimal'
↪'libpython3.8-stdlib' )
            install_deb_packages --operation install --packages packages[@]

            packages=( 'liblzma-dev' 'libghc-lzma-dev' 'liblz-dev' 'lzma-dev'
↪'libqdbm-dev' )
            install_deb_packages --operation install --packages packages[@]

            if [ "$(echo "$release" | grep '18.10')" ]; then
                packages=( 'libncurses6' 'libncurses6-dbg' 'libncursesw6'
↪'libncursesw6-dbg' )
                install_deb_packages --operation install --packages packages[@]
            fi
            return 0

        elif [ "$(grep -i ubuntu /etc/os-release)" ] || [ "$(grep -i mint /etc/os-
↪release)" ]; then

            std_error_exit "Ubuntu-based distribution of unknown family or release.␣
↪Exit" "$E_OSERROR"

        fi

        std_error_exit "OS package dependencies failed to install correctly" "$E_
↪DEPENDENCY"
        #
        # <-- end function os_packages -->
}


function package_info(){
    ##
    ##  displays information about this library module
    ##
    ##      - dependent module colors.sh is located always adjacent
    ##      - sourcing of dep modules must occur after local var to avoid overwrite
    ##        of variable values in this module
```

(continues on next page)

(continued from previous page)

```
    ##
    local version=$__version__
    local act="$(echo -e ${a_orange})"
    local rst="$(echo -e ${reset})"
    local i path clearscn
    local tmpdir='/tmp'
    local logs
    local col1
    local total_index
    local l_index  r_index

    declare -a arr_modules              # array; all file modules comprising buildpy
    declare -a arr_functions            # array; all function names
    declare -a arr_left                 # array; left-hand side column function names
    declare -a arr_right                # array; right-hand side column function names

    clearscn=$(which clear 2>/dev/null)

    # bash module ecosystem stats
    modules='1'                         # start @ 1 bc of main binary
    logs='0'
    col1='25'                           # lhs column fixed width

    # add main executable to list of bash modules (.modules file)
    printf -- '\n%-12s%s %-37s%s\n' " " "-" "${cyan}$pkg${rst}" "$pkg_path" > $tmpdir/
↪.modules

    for path in $pkg_lib/*; do

        i=${path##*/}

        if [ "${i#*.}" = "log" ] || [ "${i#*.}" = "pkg" ]; then
            continue

        else
            if [ "$(echo $i | wc -c)" -ge "15" ]; then


                width=$(( $col1 - $(echo $i | wc -c) ))

                # req keep columns vertically aligned
                printf -- '%-12s%s %-37s%-'$width's\n' " " "-" "${cyan}$i${rst}" "
↪$pkg_lib" >> $tmpdir/.modules

            else
                printf -- '%-12s%s %-37s%-'$width's\n' " " "-" "${cyan}$i${rst}" "
↪$pkg_lib" >> $tmpdir/.modules
            fi
            # incr modules
            (( modules ++ ))
        fi
    done

    # finally, add bash completion file artifact (last .modules file)
    printf -- '%-12s%s %-37s%s\n' " " "-" "${cyan}buildpy-completion.bash${rst}" "/
↪etc/bash_completion.d" >> $tmpdir/.modules

```

(continues on next page)

(continued from previous page)

```
    # log files,rm path in front
    printf -- '\n' > $tmpdir/.logs

    for i in ${LOG_FILE##*/} ${LOG_CONSOLE##*/}; do

        if [ "$(echo $i | wc -c)" -ge "15" ]; then
            # req keep columns vertically aligned
            printf -- '%-12s%s %-34s\t%s\n' ' ' "-" "${cyan}$i${rst}" "$LOG_DIR" >>
↪$tmpdir/.logs
        else
            printf -- '%-12s%s %-37s%-16s\n' ' ' "-" "${cyan}$i${rst}" "$LOG_DIR" >>
↪$tmpdir/.logs
        fi

        (( logs ++ ))

    done

    # split into 2 columns:
    arr_functions=( )
    for i in $(declare -F | awk '{print $3}'); do
        imod="- ${i}"
        arr_functions=( "${arr_functions[@]}" "${imod}" )
    done

    total_index=${#arr_functions[@]}

    # construct, display program info msg output
    cat <<EOM
    $(printf "%0.s_" {1..65})

        ${title}Build ${hic}Python3 ${title}from Source | ${hic}Linux${btext}

    $(printf -- '%-4s%-12s%-19s%s\n'  ' ' "Module Name:" ' '  "${cyan}$pkg${rst}")
    $(printf -- '%-4s%2s%-16s%s\n' ' ' "Module Version:" " "  "${act}$version${rst}")
    $(printf "%0.s_" {1..65})

    $(printf -- '%-4s%2s%-19s%s\n' ' ' "Bash Modules:" " "  "$modules")
    $(cat $tmpdir/.modules)

    $(printf -- '%-4s%2s%-22s%s\n' ' ' "Log Files:" " "  "$logs")
    $(cat $tmpdir/.logs)
    $(printf "%0.s_" {1..65})

    $(printf -- '\t%2s%-17s%s\n'  "Bash Functions:" " "  "$total_index")

EOM

    l_index=$(( $(( $total_index / 2 )) + 1 ))

    for j in "${arr_functions[@]:0:$l_index}"; do
        j_trunc=$(echo $j | cut -c 1-26)
        arr_left=( "${arr_left[@]}" "$j_trunc" )
    done

    r_index=$(( $l_index + 1 ))

```

(continues on next page)

(continued from previous page)

```
    for k in "${arr_functions[@]:$r_index:$total_index}"; do
        arr_right=( "${arr_right[@]}" "$k" )
    done

    middle_index="$l_index"

    for m in $(seq 0 $middle_index); do

        # calc the width of left-hand column element
        if [[ "${arr_left[$m]}" ]] || [[ "${arr_right[$m]}" ]]; then

            width=$(echo "${arr_left[$m]}" | wc -c)

            # subtract lhs column width - width of element; difference is rhs column offset
            col2=$(( $col1 - $width ))
            printf -- '%-12s%-'$col1's\t%-'$col2's\n' ' ' "${arr_left[$m]}" "${arr_right[$m]}"
        else
            continue
        fi
    done
    printf -- '\n'
    # clean up
    rm $tmpdir/.functions $tmpdir/.modules $tmpdir/.logs 2>/dev/null
    #
    # <<-- end function package_info -->>
}


function post_install(){
    ##
    ## prints ending message and stats ##
    ##
    local endtime="$1"

    if [ ! $endtime ]; then

        START=$(timer)
        std_message "Python3 installation start: ${title}$(date +'%H:%M:%S')${reset}" "INFO" $LOG_FILE
        return 1

    else

        std_message "Python3 installation COMPLETE at $(date +'%H:%M'). Total runtime: ${title}$(timer $endtime)${reset}." "INFO" $LOG_FILE
        std_message "See the following for installation details:\n
        \t* Log file:  ${yellow}$LOG_FILE${reset}\n
        \t* Console Event Log: ${yellow}$LOG_CONSOLE${reset}" "INFO"

    fi

    return 0
    #
    # <-- end function print stats -->
}
```

(continues on next page)

(continued from previous page)

```
2166
2167
2168 function preinstall_validation(){
2169     ##
2170     ##  Ensures version to be installed does not already exist
2171     ##
2172     local major="$1"
2173     if which "python${major}" 2>&1 >/dev/null; then
2174         std_warn "Python $major is already active on this system. Aborting␣
     ↪installation" $LOG_FILE
2175         exit 0
2176     fi
2177 }
2178
2179
2180 function root_permissions(){
2181     ##
2182     ## validates required root privileges ##
2183     ##
2184     if [ $EUID -ne 0 ]; then
2185
2186         std_warn "This command requires root or sudo access to root privileges.\n
2187         \n\t\tRe-run as root or execute with sudo:
2188             \n\t\t\t$ sudo  $pkg  <parameters>"
2189         printf -- "\n\n"
2190         std_logger "Permission denied. Rerun as root or execute with sudo" "WARN"
     ↪$LOG_FILE
2191         exit $E_DEPENDENCY
2192
2193     elif [ $SUDO_USER ]; then
2194
2195         SUDO=""
2196         logfile_owner $LOG_FILE $LOG_CONSOLE
2197
2198     fi
2199
2200     return 0
2201
2202     #
2203     # <-- end function root_permissions -->
2204 }
2205
2206
2207 function root_permissions_bool(){
2208     ##
2209     ## validates required root privileges, return bool only ##
2210     ##
2211
2212     if [ $EUID -eq 0 ] || [ $SUDO_USER ]; then
2213
2214         return 0
2215
2216     fi
2217
2218     return 1
2219
2220     #
```

(continues on next page)

(continued from previous page)

```
    # <-- end function root_permissions -->
}


function source_profile(){
    ##
    ## used to import aliases and other environment elements at runtime ##
    ##

    # source paths
    if [[ -f "$HOME/.bashrc" ]]; then

        source "$HOME/.bashrc" 2>/dev/null

    elif [[ -f "$HOME/.bash_profile" ]]; then

        source "$HOME/.bash_profile"  2>/dev/null

    fi
    return 0
    #
    # <-- end function source_profile -->
}


function test_removal(){
    ##
    ## tests to ensure python binaries and deps removed ##
    ##
    local major="$1"
    local python_ver="python$major"
    #
    std_message "Testing removal of Python $major..." "TEST" $LOG_FILE

    # delay for user read
    sleep 1

    if [ "$($python_ver --version 2>/dev/null)" ]; then

        std_warn "Problem removing Python $major. Identifying remaining file␣
↪locations:" $LOG_FILE
        updatedb -v
        std_message "$(locate $python_ver)" "INFO" $LOG_FILE

    else

        std_message "Python $major ${green}successfully${reset} removed." "INFO" $LOG_
↪FILE

    fi
    #
    # <-- end function test_removal -->
}


function uninstall_disclaimer(){
    ##
```

(continues on next page)

(continued from previous page)

```
    ##  Display disclaimer msg prior to initiating uninstall
    ##
    cat <<EOM
        ${red}_____________________________________________________________________________$
↪{reset}

                                    ${title} DISCLAIMER ${btext}
        ${red}_____________________________________________________________________________$
↪{reset}

              You are about to remove system-level python $major binaries
              and dependencies.

              Uninstalling a Python version you have installed using this
              program generally works fine. Please note that uninstalling
              a Python version included natively with your operating
              system is ${ul}not recommended${reset}.

              Depending on your Linux distribution, some operating system
              functions may depend upon the Python version(s) included
              natively with the OS.

              Although the uninstall functionality has been successfully
              tested on all supported major Linux distributions,


                              ${red}PROCEED AT YOUR OWN RISK${reset}


EOM
    return 0
    #
    # <--- end function uninstal_disclaimer --->
}


function uninstall(){
    ##
    ## uninstalls system-level python3 ##
    ##
    #
    #   This should ideally work via the following:
    #       1. find /usr -name "python3.X"
    #       2. delete paths that result
    #       3. Install and run mlocate after to find any paths left
    #
    #   Installing mlocate is OS-specific (see function mlocate_installation):
    #      REDHAT (for example, remove python3.6):
    #        sudo yum -y update
    #        # install epel
    #        curl -o epel.rpm https://dl.fedoraproject.org/pub/epel/epel-release-
↪latest-7.noarch.rpm
    #        sudo yum -y install epel.rpm
    #        sudo yum -y update
    #        sudo yum -y install mlocate
    #        sudo updatedb
    #
```

(continues on next page)

(continued from previous page)

```
    local major="$1"
    local wipe_clean="$2"
    local choice
    local python_str="Python-$major"
    local python_ver="python$major"
    declare -a arr_paths

    if ! binary_installed_bool "python$major"; then
        msg="You are attempting to uninstall $python_str, which is not installed. Exit
↪"

        if [ $QUIET ]; then
            std_logger "$msg" "WARN" $LOG_FILE
        else
            std_error_exit "$msg" $E_DEPENDENCY
        fi
    fi

    # print disclaimer msg
    msg="Initiate uninstall of Python $major binaries"

    if [ $QUIET ]; then
        std_logger "$msg" "INFO" $LOG_FILE
    else
        uninstall_disclaimer
        # choose
        if continue_on; then
            std_logger "$msg" "INFO" $LOG_FILE
        else
            # abort uninstall
            printf -- '\n'; exit 0
        fi
    fi

    # create backup of pip packages
    pip_backup

    # display paths found
    std_message "Found paths to delete:" "INFO" $LOG_FILE "pprint"

    mapfile -t arr_paths < <(find /usr/local -name "$python_ver" 2>/dev/null)

    for path in "${arr_paths[@]}"; do
        std_message "\t$path" "INFO" $LOG_FILE "pprint"
    done

    std_message 'Please review the uninstall paths' 'INFO' $LOG_FILE
    if ! continue_on; then return 0; fi

    # remove python3 version ("python_ver")
    for path in "${arr_paths[@]}"; do
        std_message "Deleting $path" "INFO" $LOG_FILE
        rm -fr "$path"
        printf -- "\n"
    done

    # check specific paths noted in paths filemap
```

(continues on next page)

(continued from previous page)

```
    for path in $(cat $pkg_lib/uninstall.paths); do
        std_message "Deleting $path" "INFO" $LOG_FILE
        rm -fr "${path:?}/$python_ver*"
        printf -- "\n"
    done

    # locate any residual locations only if WIPE CLEAN explicit option given
    if mlocate_installation && $wipe_clean; then

        for path in $(locate $python_ver); do
            std_message "Deleting path: ${yellow}$path${btext}" "INFO" $LOG_FILE
            rm -fr $path
            printf -- "\n"
        done

    fi

    return 0
    #
    # <-- end function uninstall -->
}


function update_path(){
    ##
    ## Update PATH environment variable:
    ##       - /usr/local/bin
    ##

    # ensure /usr/local/bin for python executables in PATH
    if [ ! "$(echo $PATH | grep '\/usr\/local\/bin')" ]; then

        if [ -f $HOME/.bashrc ]; then

            printf -- '%s\n' "PATH=\$PATH:/usr/local/bin" >> "$HOME/.bashrc"
            printf -- '%s\n' "export PATH" >> "$HOME/.bashrc"

        elif [ -f "$HOME/.bash_profile" ]; then

            printf -- '%s\n' "PATH=\$PATH:/usr/local/bin" >> "$HOME/.bash_profile"
            printf -- '%s\n' "export PATH" >> "$HOME/.bash_profile"

        elif [ -f "$HOME/.profile" ]; then

            printf -- '%s\n' "PATH=\$PATH:/usr/local/bin" >> "$HOME/.profile"
            printf -- '%s\n' "export PATH" >> "$HOME/.profile"

        fi

    fi

    # ensure local bin for python packages in PATH
    if [ ! "$(echo $PATH | grep $HOME\/.local\/bin)" ]; then

        if [ -f $HOME/.bashrc ]; then

            printf -- '%s\n' "PATH=\$PATH:$HOME/.local/bin" >> "$HOME/.bashrc"
```

(continues on next page)

(continued from previous page)

```
            printf -- '%s\n' "export PATH" >> "$HOME/.bashrc"

        elif [ -f "$HOME/.bash_profile" ]; then

            printf -- '%s\n' "PATH=\$PATH:$HOME/.local/bin" >> "$HOME/.bash_profile"
            printf -- '%s\n' "export PATH" >> "$HOME/.bash_profile"

        elif [ -f "$HOME/.profile" ]; then

            printf -- '%s\n' "PATH=\$PATH:$HOME/.local/bin" >> "$HOME/.profile"
            printf -- '%s\n' "export PATH" >> "$HOME/.profile"

        fi

    fi

    source_profile
    return 0
    #
    # <-- end function update_path -->
}


function upgrade_pip(){
    ##
    ##  - upgrades pip3, setuptools
    ##  - sets symlink to pip for python3
    ##  - multiple update cycles to accommodate pip 9.x > pip 10.x > pip 18.x
    ##
    local pip_bin
    local count="0"
    #
    pip_bin="$(which pip3 2>/dev/null)"
    source_profile

    # set path to pip binary
    if [ "$pip_bin" ]; then

        std_message "Found pip3 at path: ${yellow}$pip_bin${reset}" "INFO" $LOG_FILE

    else
        if [ -f "/usr/local/bin/pip3" ]; then
            pip_bin="/usr/local/bin/pip3"

        elif [ "$(which pip 2>/dev/null)" ]; then
            pip_bin="$(which pip)"

        elif [ -f "/usr/local/bin/pip" ]; then
            pip_bin="/usr/local/bin/pip"

        else
            std_warn "Skipping pip / pip3 upgrade -- pip executable could not be found
↪" $LOG_FILE
            return 1
        fi
    fi

```

(continues on next page)

(continued from previous page)

```
    #
    if [ $SUDO_USER ]; then
        # user using sudo
        while (( $count <= 2 )); do

            # check for latest version, pip
            if [ "$($pip_bin list --outdated  2>/dev/null | grep pip)" ]; then

                std_message "Upgrade pip3 to latest, set python3 symlink for pip" "INFO" $LOG_FILE
                $pip_bin install -U pip  >> $LOG_CONSOLE 2>/dev/null &
                progress_dots --text "Upgrading pip3"

            else
                std_message "pip $($pip_bin --version 2>/dev/null | awk '{print $2}') installed - latest" "INFO" $LOG_FILE
                break
            fi

            # check for latest version, setuptools
            if [ "$($pip_bin list --outdated 2>/dev/null | grep setuptools)" ]; then

                std_message "Upgrade setuptools to latest" "INFO" $LOG_FILE
                $pip_bin install -U setuptools >> $LOG_CONSOLE  2>/dev/null &
                progress_dots --text "upgrading setuptools"

            else
                std_message "setuptools $($pip_bin list | grep setuptools | awk '{print $2}') installed - latest" "INFO" $LOG_FILE
            fi

            count=$(( $count + 1 ))

        done

    elif [ "$pip_bin" ]; then
        # root user
        while (( $count <= 2 )); do

            # check for latest version, pip
            if [ "$($pip_bin list --outdated  2>/dev/null | grep pip)" ]; then

                std_message "Upgrade pip3 to latest, set python3 symlink for pip" "INFO" $LOG_FILE
                $pip_bin install -U pip  >> $LOG_CONSOLE 2>/dev/null &
                progress_dots --text "Upgrading pip3"

            else
                std_message "pip $($pip_bin --version 2>/dev/null | awk '{print $2}') installed - latest" "INFO" $LOG_FILE
                break
            fi

            # check for latest version, setuptools
            if [ "$($pip_bin list --outdated 2>/dev/null | grep setuptools)" ]; then

                std_message "Upgrade setuptools to latest" "INFO" $LOG_FILE
```

(continues on next page)

(continued from previous page)

```
                $pip_bin install -U setuptools >> $LOG_CONSOLE  2>/dev/null &
                progress_dots --text "upgrading setuptools"

            else
                std_message "setuptools $($pip_bin list | grep setuptools | awk '{print $2}') installed - latest" "INFO" $LOG_FILE
            fi

            count=$(( $count + 1 ))

        done

    else

        std_warn "Unable to upgrade pip3, setuptools. Please use manual upgrade" $LOG_FILE

    fi
    #
    # <-- end function upgrade_pip -->
}


function verify_rc_status(){
    ##
    ##  Reset VERSION global variable in case binary set to be
    ##  compiled is a release candidate, alpha, or beta code.
    ##
    local version="$1"
    if [[ $(ls $TMPDIR | grep "Python-$version.tar.xz") ]]; then
        return 0

    elif [[ $(ls $TMPDIR | grep "Python-${version}rc[1-4].tar.xz") ]]; then
        bin=$(ls $TMPDIR | grep "Python-${version}rc[1-4].tar.xz")
        std_warn "Release Candidate binary ($bin). Stable version NOT available" $LOG_FILE
        if continue_on "Do you want to install anyway?"; then
            return 0
        else
            exit 0
        fi

    elif [[ $(ls $TMPDIR | grep "Python-$version[a-b][1-4].tar.xz") ]]; then
        bin=$(ls $TMPDIR | grep "Python-${version}[a-b][1-4].tar.xz")
        std_warn "Beta or Alpha code binary ($bin)." $LOG_FILE
        if continue_on "Do you want to install anyway?"; then
            return 0
        else
            exit 0
        fi
    fi
}


function clean_up(){
    ##
    ## remove build artifacts ##
```

(continues on next page)

(continued from previous page)

```
    ##
    local version="$1"                  # full version (X.Y.Z) to remove
    local yl=$(echo -e ${yellow})       # yellow link color code
    local clean_status="0"              # return code container
    local tmpdir='/tmp'
    declare -a arr_clean

    ##   remove build artifacts for a specific version   ##
    if [ $version ]; then

        mapfile -t arr_clean < <(find /tmp -name "Python-${version}*" 2>/dev/null)

        std_message "Begin build artifict removal for Python $version only..." "INFO"
↪$LOG_FILE
        for residual in "${arr_clean[@]}"; do

            for artifact in $residual; do
                # .tar, .tar.xz, or dir object
                rm -fr $artifact

                # success | failure status
                if [ -e "$artifact" ] || [ -f "$artifact" ] || [ -d "$artifact" ] ;␣
↪then
                    std_warn "Unable to remove ${yl}${artifact}${reset}" $LOG_FILE
                    clean_status="1"        # something went wrong
                else
                    std_message "Successfully removed ${yl}${artifact}${reset}" "OK"
↪$LOG_FILE
                fi
            done
        done
        return $clean_status

    else
        ##   remove build artifacts for all versions   ##
        find $tmpdir -name "Python-*" 2>/dev/null > $tmpdir/.s.results

        for i in $(cat $tmpdir/.s.results); do
            arr_clean=(  "${arr_clean[@]}" $i  )
        done

        if [ -z "${arr_clean[*]}" ]; then
            # nothing to clean
            std_warn "No build artifacts found to remove" $LOG_FILE
            return $clean_status
        fi

        std_message "Locating all local build artificts..." "INFO" $LOG_FILE

        for residual in "${arr_clean[@]}"; do

            rm -fr $residual

            # success | failure status
            if [ -e "$residual" ] || [ -f "$residual" ] || [ -d "$residual" ] ; then
                std_warn "Unable to remove ${yl}${residual}${reset}" $LOG_FILE
                clean_status="1"        # something went wrong
```

(continues on next page)

(continued from previous page)

```
            else
                std_message "Successfully removed ${yl}${residual}${reset}" "OK" $LOG_
↪FILE
            fi

        done
        return $clean_status
    fi
    #
    # <-- end function clean_up -->
}


function show_downloads_display(){
    ##
    ##
    ##
    local downloads location major size i
    local tmpfile sorted_tmp twidth

    # set tempfile location
    if [ -d '/dev/shm' ]; then
        tmpfile='/dev/shm/downloads.tmp'
        sorted_tmp='/dev/shm/sorted.tmp'
    else
        tmpfile='/tmp/downloads.tmp'
        sorted_tmp='/tmp/sorted.tmp'
    fi

    # find and sort downloaded python binaries on local filesystem
    downloads=$(_current_downloads)

    if [ ! "$downloads" ]; then
        std_message "No current downloads" "INFO" $LOG_FILE

    else
        twidth='52'
        printf -- '\n\t   %s   %s  %s   %s   %s\n' "Version" '|' "Size" '|'
↪"Filesystem Location"
        printf '\t%*s\n' "$twidth" '' | tr ' ' -

        for i in $downloads; do
            location=$(find /tmp -name "*${i}*" 2>/dev/null)
            major=${i#Python-}
            rsize=$(ls -l --block-size=MB $location | awk '{print $5}')
            size="${rsize%MB} MB"
            printf -- '\t%s %s %s %s %s\n' "${BOLD}Python-${btext}${bbc}$major${btext}
↪" '|' "$size" '|' "$location" >> $tmpfile
        done

        # sort
        cat $tmpfile | sort -t . > $sorted_tmp
        awk  '{ printf "%-8s %-41s %-2s %2s %-3s %-2s %-30s \n", " ", $1, $2, $3, $4,
↪$5, $6}' $sorted_tmp
        printf -- '\n'
        rm  $tmpfile $sorted_tmp || true
    fi
```

(continues on next page)

(continued from previous page)

```
}


function major_version_reconcilliation(){
    ##
    ##  Test if MAJOR_VERSION includes minor version
    ##  If yes, reset MAJOR_VERSION parameter
    ##
    local parameter_version="$1"
    local major

    if [ "$(echo ${#parameter_version})" -ge 4 ]; then
        major=$(echo "$parameter_version" | awk -F '.' '{print $1"."$2}')
        echo "$major"
    else
        echo "$parameter_version"
    fi
    #
    # <-- end function major_version_reconcilliation -->
}

function version_reconcilliation(){
    ##
    ##  Test if VERSION includes minor version
    ##  If yes, reset VERSION parameter
    ##
    local parameter_version="$1"
    local major

    if [ "$(echo ${#parameter_version})" -ge 4 ]; then
        echo "$parameter_version"
    else
        echo ""
    fi
    #
    # <-- end function version_reconcilliation -->
}


# --- main -------------------------------------------------------------------


depcheck "$LOG_DIR" "$LOG_FILE" "$LOG_CONSOLE"

parse_parameters "$@"


if [ $PKG_INFO ]; then

    # display information about this program
    package_info | more

elif [ $DOWNLOAD_ONLY ]; then

    # discover latest python version available
    if latest_version_available $MAJOR_VERSION; then
        # download build assets
```

(continues on next page)

(continued from previous page)

```
        download_binary $VERSION $MAJOR_VERSION
    fi

elif [ "$OPERATION" = "SHOW" ]; then

    case "$SUBCMD" in
        'downloads')
            show_downloads_display
            exit 0
            ;;

        'os-packages')
            os_type
            SHOW_OSPACKAGES="true"
            if [ $alternate_os ]; then
                # match current os to $alternate_os; if match, os_packages '3' & set alternative os=""
                # if different os, set family and release values
                case ${alternate_os#[0-9]*} in
                    $family)
                        release=$(determine_osrelease $alternate_os)
                        alternate_os=""
                        ;;
                    *)
                        release=$(determine_osrelease $alternate_os)
                        ;;
                esac
            fi
            os_packages '3'
            printf -- '\n'
            ;;

        'Python-'[0-9].[0-9] | 'Python-'[0-9].[0-9][0-9] | [0-9].[0-9] | [0-9].[0-9][0-9])
            # convert to major version number
            var="$2"; parameter=${var#Python-}
            if is_float "$parameter"; then
                MAJOR_VERSION="$parameter"
            else
                std_error_exit "You must provide Python major version (Example: 3.7)"
            fi
            # discover latest python version available
            latest_version_available $MAJOR_VERSION
            ;;
    esac

elif [ $OS_DETECT ]; then

    # detect operating system type (test)
    os_type

elif [ "$INSTALL_OS_DEPS" ]; then
    function install_status_display(){
        # show installed status of packages
        SHOW_OSPACKAGES="true"
        os_packages
        printf -- '\n\n'
```

(continues on next page)

(continued from previous page)

```
    }

    clear
    # determine os type
    os_type
    install_status_display
    read -p "    Install all Python 3 operating system package dependencies?  [quit]: " choice
    case $choice in
        "" | "No" | "no" | "q" | "quit" | "Q*")
            printf -- '\n\n'
            exit 0
            ;;
        "y" | "Y" | "yes" | "Yes")
            # ensure root-level permissions
            root_permissions
            # install python os dependencies
            SHOW_OSPACKAGES=""      # turn off display-only variable to install
            os_packages
            ;;
        *)
            printf -- '\n\n'
            exit 0
            ;;
    esac
    printf -- '\n\t\t\t%s\n' 'FINAL OS PACKAGE INSTALLATION STATUS'
    install_status_display

elif [ $PIP_BACKUP ]; then

    # update pip / pip3 and core deps to latest if root
    if root_permissions_bool; then
        upgrade_pip
    fi

    # create backup of pip packages
    pip_backup

elif [ $CLEAN_UP ]; then

    root_permissions

    if [ "$CLEAN_VERSION" = 'ALL' ]; then
        clean_up

    # discover latest python version available
    elif latest_version_available $MAJOR_VERSION; then
        # rm build artifacts
        clean_up $VERSION
    fi

elif [ $INSTALL ]; then

    # start timer
    START=$(timer)

    # validate root privileges
```

(continues on next page)

(continued from previous page)

```
    root_permissions

    # reset version vars if MAJOR_VERION contains full version (major.minor)
    VERSION=$(version_reconcilliation $MAJOR_VERSION)
    MAJOR_VERSION=$(major_version_reconcilliation $MAJOR_VERSION)

    # ensure requested python binaries not currently installed
    preinstall_validation $MAJOR_VERSION

    # discover latest python version available
    latest_version_available $MAJOR_VERSION || exit $E_DEPENDENCY

    std_message "Major Python version marked for installation:  ${pv_blue}$MAJOR_
↪VERSION${reset}\n \
         Full Python version marked for installation: ${bbw}$VERSION${reset}" "INFO"
    sleep 3

    # create backup of pip packages
    pip_backup

    # determine os type
    os_type

    # download build assets
    download_binary $VERSION $MAJOR_VERSION
    verify_rc_status $VERSION

    # install required operating system packages
    os_packages $VERSION

    ## --- unpack --- ##
    cd $TMPDIR || exit $E_OSERROR
    std_message "Decompressing binary... " "INFO"  $LOG_FILE
    VERSION="$(reset_rc_version $VERSION)"
    unxz Python-$VERSION.tar.xz  2>&1 | tee $tty >> $LOG_CONSOLE
    tar -xvf Python-$VERSION.tar  2>&1 | tee $tty >> $LOG_CONSOLE

    ## --- pre-build --- ##
    std_message "Begin Python-$VERSION Configure Stage..." "INFO" $LOG_FILE
    cd $TMPDIR/Python-$VERSION || exit $E_OSERROR
    configure_python "$OPTIMIZATIONS" "$QUIET"
    std_message "End Python-$VERSION Configure Stage..." "OK" $LOG_FILE

    ## --- build, install --- ##
    std_message "Begin Python-$VERSION compile with make..." "INFO" $LOG_FILE
    make -j $PARALLEL_JOB_CT   2>&1  |  tee $tty >> $LOG_CONSOLE
    std_message "Compilation stage completed successfully" "OK" $LOG_FILE
    std_message "Begin Python-$VERSION install..." "INFO" $LOG_FILE
    make install  2>&1 | tee $tty >> $LOG_CONSOLE
    std_message "Python-$VERSION Installation completed successfully..." "OK" $LOG_
↪FILE

    # clean up build artifacts
    std_message "Clean up -- remove build artifacts" "INFO" $LOG_FILE
    clean_up

    # If missing, create python3 global symlink
```

(continues on next page)

(continued from previous page)

```
    create_global_symlink "$MAJOR_VERSION" "$LOG_FILE"

    # update pip3, install symlink to pip fro python3
    upgrade_pip

    # print ending runtime stats
    post_install $START

elif [ $UNINSTALL ]; then

    # uninstall Python 3.X Version
    if [ ! $UNINSTALL_VERSION ]; then

        std_error_exit "You must provide a Python3 major version number to remove\n
        \tExample: \n\n\t\t    $ sudo ./buildpy --uninstall ${title}3.6${reset}"

    else

        # validate root privileges
        root_permissions

        # uninstall binaries
        uninstall "$UNINSTALL_VERSION" "$WIPE_CLEAN"

        # validate removal
        test_removal "$UNINSTALL_VERSION"

    fi

elif [ $DISPLAY_VERSION ]; then

    display_program_version

fi


# <-- end -->
exit 0
```

Back to *buildpy*

Back to *buildpy Source Code*

24.3 colors.sh

Back to *Bash Module Index*

```
#!/usr/bin/env bash

pkg=$(basename $0 2>/dev/null)

#-------------------------------------------------------------------------------
#
#   colors.sh module | std colors for bash
#
#-------------------------------------------------------------------------------
#   Bright ansi color codes:
#       Bright Black: \u001b[30;1m
#       Bright Red: \u001b[31;1m
#       Bright Green: \u001b[32;1m
#       Bright Yellow: \u001b[33;1m
#       Bright Blue: \u001b[34;1m
#       Bright Magenta: \u001b[35;1m
#       Bright Cyan: \u001b[36;1m
#       Bright White: \u001b[37;1m
#
#-------------------------------------------------------------------------------

VERSION="2.0.6"


# --- standard bash color codes  ------------------------------------------------
↪-------------


    # std color codes
    red=$(tput setaf 1)
    green=$(tput setaf 2)
    yellow=$(tput setaf 3)
    blue=$(tput setaf 4)
    purple=$(tput setaf 5)
    cyan=$(tput setaf 6)
    white=$(tput setaf 7)
    gray=$(tput setaf 008)
    magenta=$(tput setaf 13)

    # Formatting
    BOLD=`tput bold`
    UNBOLD=`tput sgr0`
    a_italic='\x1b[3m'
    ITALIC=$(echo -e ${a_italic})

    # std reset
    reset=$(tput sgr0)


# --- ansi color escape codes  --------------------------------------------------
↪-------------

    # ansi color codes
    a_bluegray='\033[38;5;68;38;5;68m'
    a_darkblue='\033[34;5;21m'
    a_dgray='\033[38;5;95;38;5;8m'                  # dark gray
    a_lgray='\033[38;5;95;38;5;245m'                # light gray
```

(continues on next page)

(continued from previous page)

```
    a_magenta='\033[38;5;95;38;5;177m'
    a_orange='\033[38;5;95;38;5;214m'
    a_wgray='\033[38;5;95;38;5;250m'                    # white-gray
    a_wgray2='\033[38;5;7m;38;5;250m'                           # whitest-gray

    # ansi bright colors
    a_brightblue='\033[38;5;51m'
    a_brightcyan='\033[38;5;36m'
    a_brightgreen='\033[38;5;95;38;5;46m'
    a_bluepurple='\033[38;5;68m'
    a_brightred='\u001b[31;1m'
    a_brightyellow='\033[38;5;11m'
    a_brightyellow2='\033[38;5;95;38;5;226m'
    a_brightyellowgreen='\033[38;5;95;38;5;155m'
    a_brightwhite='\033[38;5;15m'

    # ansi font formatting
    bold='\u001b[1m'                                    # ansi format
    underline='\u001b[4m'                               # ansi format

    # ansi escape code reset
    resetansi='\u001b[0m'


# --- color print variables  ----------------------------------------------------
--------------



    # Initialize ansi colors
    title=$(echo -e ${bold}${white})
    url=$(echo -e ${underline}${a_brightblue})
    options=$(echo -e ${white})
    commands=$(echo -e ${a_brightcyan})                 # use for ansi escape color␣
↪codes

    # frame codes (use for tables)                  SYNTAX:  color:format (bold, etc)
    pv_blue=$(echo -e ${a_brightblue})
    pv_bluebold=$(echo -e ${bold}${a_brightblue})
    pv_green=$(echo -e ${a_brightgreen})                # use for tables; green border␣
↪faming
    pv_greenbold=$(echo -e ${bold}${a_brightgreen})     # use for tables; green bold␣
↪border faming
    pv_orange=$(echo -e ${a_orange})                    # use for tables; orange border␣
↪faming
    pv_orangebold=$(echo -e ${bold}${a_orange})         # use for tables; orange bold␣
↪border faming
    pv_white=$(echo -e ${a_brightwhite})                # use for tables; white border␣
↪faming
    pv_whitebold=$(echo -e ${bold}${a_brightwhite})     # use for tables; white bold␣
↪border faming

    pv_bodytext=$(echo -e ${reset}${a_wgray})           # main body text; set to reset␣
↪for native xterm
    pv_bg=$(echo -e ${a_brightgreen})                   # brightgreen foreground cmd
    pv_bgb=$(echo -e ${bold}${a_brightgreen})           # bold brightgreen foreground␣
↪cmd
    pv_wgray=$(echo -e ${a_wgray})
```

(continues on next page)

(continued from previous page)

```
    pv_orange=$(echo -e ${a_orange})
    pv_wgray=$(echo -e ${a_wgray})
    pv_lgray=$(echo -e ${a_lgray})
    pv_dgray=$(echo -e ${a_dgray})

    # initialize default color scheme
    accent=$(tput setaf 008)                        # ansi format
    ansi_orange=$(echo -e ${a_orange})              # use for ansi escape color codes

    # reset print variable
    RESET=$(echo -e ${resetansi})


# --- declarations  ------------------------------------------------------------

    # indent, x spaces
    function indent02() { sed 's/^/  /'; }
    function indent04() { sed 's/^/    /'; }
    function indent10() { sed 's/^/          /'; }
    function indent15() { sed 's/^/               /'; }
    function indent18() { sed 's/^/                  /'; }
    function indent20() { sed 's/^/                    /'; }
    function indent25() { sed 's/^/                         /'; }
    function indent40() { sed 's/^/                                        /'; }


# --- aliases  -----------------------------------------------------------------

    # alias for legacy backard compatibility
    alias orange=$a_orange
    alias wgray=$a_wgray
    alias lgray=$a_lgray
    alias dgray=$a_dgray
    alias bodytext=$pv_bodytext
    alias blue_frame=$pv_blue
    alias bluebold_frame=$pv_bluebold
    alias green_frame=$pv_green
    alias greenbold_frame=$pv_greenbold
    alias orange_frame=$pv_orange
    alias orangebold_frame=$pv_orangebold
    alias white_frame=$pv_white
    alias whitebold_frame=$pv_whitebold
    alias brightblue=$a_brightblue
    alias brightcyan=$a_brightcyan
    alias brightgreen=$a_brightgreen
    alias bluepurple=$a_bluepurple
    alias brightred=$a_brightred
    alias brightyellow=$a_brightyellow
    alias brightyellow2=$a_brightyellow2
    alias brightyellowgreen=$a_brightyellowgreen
    alias brightwhite=$a_brightwhite


```

(continues on next page)

(continued from previous page)

```
# --- display about  ----------------------------------------------------------
↪--------------


function print_local_variables(){
    # print out all variables contained in this module:
    VARS=$(set -o posix ; set)
    SCRIPT_VARS=$(grep -vFe "$VARS" <<<"$(set -o posix ; set)" | grep -v ^VARS=)
    echo -e "$SCRIPT_VARS"
    unset VARS
}


function print_colors(){
    # print out all variables contained in this module:
    declare -a array=("${!1}")
    local rst=$(echo -e ${reset})

    for i in "${array[@]}"; do
        var="$(echo -e ${i})"
        printf -- '\t%s\n' $var
    done
    return 0
}


function pkg_info(){
    ##
    ##  displays information about this library module
    ##
    ##      - dependent module colors.sh is located always adjacent
    ##      - sourcing of dep modules must occur after local var to avoid overwrite
    ##        of variable values in this module
    ##
    local version="$1"
    local bdwt=$(echo -e ${bold}${a_brightwhite})
    local act=$(echo -e ${a_orange})
    local rst=$(echo -e ${reset})

    declare -a ansi_colors
    declare -a printvalue_colors

    # generate list of functions
    printf -- '%s\n' "$(declare -F | awk '{print $3}')" > /tmp/.functions
    sum=$(cat /tmp/.functions | wc -l)

    # construct, display help msg output
    cat <<EOM
    ____________________________________________________

    ${title}colors.sh${rst}:  Bash Color Library

    Module Name:        ${cyan}$pkg${rst}
    Module Version:     ${act}$version${rst}
    ____________________________________________________

    Module Contains $sum Functions:
```

(continues on next page)

(continued from previous page)

```

EOM
    # display list of function names in this module
    for l in $(cat /tmp/.functions); do
        printf -- '\t%s %s\n' "-" "$1"
    done
    printf -- '\n'
    rm /tmp/.functions

    # show vars contained
    ansi_colors=$(set -o posix ; set | grep 'a_')
    asum=$(set -o posix ; set | grep 'a_' | wc -l)

    printvalue_colors=$(set -o posix ; set | grep 'pv_')
    pvsum=$(set -o posix ; set | grep 'pv_' | wc -l)

    #  display color vars
    cat <<EOM
    ____________________________________________________

    ${rst}ANSI Codes:  $asum

    $(print_colors ansi_colors[@])

    ${rst}Print Value Codes:  $pvsum

    $(print_colors printvalue_colors[@])
    ${rst}

EOM
    #
    # <<-- end function pkg_info -->>
}

    # print information about this package
    if [ "$pkg" = "colors.sh" ]; then
        pkg_info "$VERSION"
        exit 0
    fi
```

Back to *Bash Module Index*

24.4 exitcodes.sh

```
#!/usr/bin/env bash

#
#   EXIT ERROR CODES -- source via script
#       ERROR CODES, version 1.3
#

# error codes
E_OK='0'                    # exit code if normal exit conditions
```

(continues on next page)

(continued from previous page)

```
E_DEPENDENCY='1'          # exit code if missing required ec2cdependency
E_PERMISSIONS='2'         # exit code if inadequate permissions for attempted␣
↪operation
E_NOLOG='3'               # exit code if failure to create log dir, log file
E_BADSHELL='4'            # exit code if incorrect shell detected
E_AUTH='5'                # exit code if authentication failure
E_CONFIG='6'              # exit code if configuration file missing or corrupted
E_OSERROR='7'             # exit code if fail to identify os or os-specific attribute
E_USER_CANCEL='8'         # exit code if user cancel
E_BADARG='9'              # exit code if bad input parameter
E_NETWORK_ACCESS='10'     # exit code if no network access from current location
E_EXPIRED_CREDS='11'      # exit code if temporary credentials no longer valid
E_MISC='31'               # exit code if miscellaneous (unspecified) error
```

Back to *Bash Module Index*

24.5 version.py

```
__version__="1.8.12"
```

Back to *Bash Module Index*

24.6 os_distro.sh

Back to *Bash Module Index*

```
#!/usr/bin/env bash

#
#   Author:                 based on the original by zeldarealm@gmail.com
#   Source:                 https://github.com/KittyKatt/screenFetch
#        Instructions:       source function-library/os_distro.sh ; detectdistro
#        Version:               1.4.2

verboseOut () {
        if [[ "$verbosity" -eq "1" ]]; then
                printf "\033[1;31m:: \033[0m$1\n"
        fi
}

errorOut () {
        printf "\033[1;37m[[ \033[1;31m! \033[1;37m]] \033[0m$1\n"
}

```

(continues on next page)

(continued from previous page)

```
stderrOut () {
        while IFS='' read -r line; do printf "\033[1;37m[[ \033[1;31m! \033[1;37m]] \033[0m${line}\n"; done
}


function amazonlinux_release_version(){
        #
        #  determines release version internally from  within an
        #  amazonlinux host os environment.
        #
        #  Requires identification of AmazonLinux OS Family as a
        #  prerequisite
        #
        local image_id
        local region
        local cwd=$PWD
        local tmp='/tmp'
        #
        cd $tmp
        curl -O 'http://169.254.169.254/latest/dynamic/instance-identity/document'
        image_id="$(jq -r .imageId $tmp/document)"
        region="$(jq -r .region $tmp/document)"
        aws ec2 describe-images --image-ids $image_id --region $region > $tmp/images.json
        printf -- "%s\n" "$(jq -r .Images[0].Name $tmp/images.json | awk -F '-' '{print $1}')"
        rm $tmp/document $tmp/images.json
        cd $cwd
}


function fedora_release_version(){
        #
        #  determines Redhat release version internally from
        #  within a Redhat host os environment.
        #
        #  Requires identification of Redhat Linux OS Family
        #  as a prerequisite
        #
        local release
        #
        release="$(grep VERSION_ID /etc/os-release | awk -F '=' '{print $2}')"
        printf -- "%s\n" "$(echo $release)"
}


function fedora_codename_version(){
        #
        #  determines Redhat codename internally from
        #  within a Redhat host os environment.
        #
        #  Requires identification of Redhat Linux OS Family
        #  as a prerequisite
        #
        local codename

```

(continues on next page)

(continued from previous page)

```
        codename="$(grep CODENAME /etc/os-release | awk -F '=' '{print $2}')"

        if [ "$codename" != \"\" ]; then
                printf -- '%s\n' $codename
        else
                printf -- '%s\n' ""
        fi
}


function redhat_release_version(){
        #
        #  determines Redhat release version internally from
        #  within a Redhat host os environment.
        #
        #  Requires identification of Redhat Linux OS Family
        #  as a prerequisite
        #
        local release
        #
        release="$(grep VERSION_ID /etc/os-release | awk -F '=' '{print $2}')"
        printf -- "%s\n" $(echo $release | cut -c 2-50 | rev | cut -c 2-50 | rev)
}


function redhat_codename_version(){
        #
        #  determines Redhat codename internally from
        #  within a Redhat host os environment.
        #
        #  Requires identification of Redhat Linux OS Family
        #  as a prerequisite
        #
        local codename
        codename="$(grep VERSION /etc/os-release | head -n 1 | awk -F '(' '{print $2}
↪' | awk -F ')' '{print $1}')"
        printf -- "%s\n" $codename
}


detectdistro () {
        ##␣
↪
↪#
        #␣
↪         determines:
↪#
        #                - os␣
↪family
↪#
        #                - release (revision)␣
↪number                                                  #
        #                - codename (informaal␣
↪name)                                                  #
        ##
        local format="$1"       #  accepts "json" or '' (None)
        #
```

(continues on next page)

(continued from previous page)

```
        if [[ -z "${distro}" ]]; then
                distro="Unknown"
                # LSB Release Check
                if type -p lsb_release >/dev/null 2>&1; then
                        # read distro_detect distro_release distro_codename <<< $(lsb_release -sirc)
                        distro_detect=( $(lsb_release -sirc) )
                        if [[ ${#distro_detect[@]} -eq 3 ]]; then
                                distro_codename=${distro_detect[2]}
                                distro_release=${distro_detect[1]}
                                distro_detect=${distro_detect[0]}
                        else
                                for ((i=0; i<${#distro_detect[@]}; i++)); do
                                        if [[ ${distro_detect[$i]} =~ ^[[:digit:]]+((.[[:digit:]]+|[[:digit:]]+|)+)$ ]]; then
                                                distro_release=${distro_detect[$i]}
                                                distro_codename=${distro_detect[@]:$(($i+1)):${#distro_detect[@]}+1}
                                                distro_detect=${distro_detect[@]:0:$i}
                                                break 1
                                        elif [[ ${distro_detect[$i]} =~ [Nn]/[Aa] || ${distro_detect[$i]} == "rolling" ]]; then
                                                distro_release=${distro_detect[$i]}
                                                distro_codename=${distro_detect[@]:$(($i+1)):${#distro_detect[@]}+1}
                                                distro_detect=${distro_detect[@]:0:$i}
                                                break 1
                                        fi
                                done
                        fi
                        case "${distro_detect}" in
                                "CentOS"|"Chapeau"|"Deepin"|"Devuan"|"DesaOS"|"Fedora"|"gNewSense"|"Jiyuu Linux"|"Kogaion"|"Korora"|"Mageia"|"Netrunner"|"NixOS"|"Pardus"|"Raspbian"|"Sabayon"|"Solus"|"SteamOS"|"Trisquel"|"Ubuntu"|"GrombyangOS"|"Scientific Linux")
                                        # no need to fix $distro/$distro_codename/$distro_release
                                        distro="${distro_detect}"
                                        ;;
                                "archlinux"|"Arch Linux"|"arch"|"Arch"|"archarm")
                                        distro="Arch Linux"
                                        distro_release="n/a"
                                        if [ -f /etc/os-release ]; then
                                                os_release="/etc/os-release";
                                        elif [ -f /usr/lib/os-release ]; then
                                                os_release="/usr/lib/os-release";
                                        fi
                                        if [[ -n ${os_release} ]]; then
                                                if grep -q 'antergos' /etc/os-release; then
                                                        distro="Antergos"
                                                        distro_release="n/a"
                                                fi
                                                if grep -q -i 'logos' /etc/os-release; then
```

(continues on next page)

(continued from previous page)

```
                                                distro="Logos"
                                                distro_release="n/a"
                                        fi
                                        if grep -q -i 'swagarch' /etc/os-
↪release; then
                                                distro="SwagArch"
                                                distro_release="n/a"
                                        fi
                                        if grep -q -i 'obrevenge' /etc/os-
↪release; then
                                                distro="OBRevenge"
                                                distro_release="n/a"
                                        fi
                                fi
                                ;;
                        "ALDOS"|"Aldos")
                                distro="ALDOS"
                                ;;
                        "artixlinux"|"Artix Linux"|"artix"|"Artix"|"Artix␣
↪release")
                                distro="Artix"
                                ;;
                        "blackPantherOS"|"blackPanther"|"blackpanther"|
↪"blackpantheros")
                                distro=$(source /etc/lsb-release; echo "
↪$DISTRIB_ID")
                                distro_release=$(source /etc/lsb-release;␣
↪echo "$DISTRIB_RELEASE")
                                distro_codename=$(source /etc/lsb-release;␣
↪echo "$DISTRIB_CODENAME")
                                ;;
                        "BLAG")
                                distro="BLAG"
                                distro_more="$(head -n1 /etc/fedora-release)"
                                ;;
                        "Chakra")
                                distro="Chakra"
                                distro_release=""
                                ;;
                        "BunsenLabs")
                                distro=$(source /etc/lsb-release; echo "
↪$DISTRIB_ID")
                                distro_release=$(source /etc/lsb-release;␣
↪echo "$DISTRIB_RELEASE")
                                distro_codename=$(source /etc/lsb-release;␣
↪echo "$DISTRIB_CODENAME")
                                ;;
                        "Debian")
                                if [[ -f /etc/crunchbang-lsb-release || -f /
↪etc/lsb-release-crunchbang ]]; then
                                        distro="CrunchBang"
                                        distro_release=$(awk -F'=' '/^DISTRIB_
↪RELEASE=/ {print $2}' /etc/lsb-release-crunchbang)
                                        distro_codename=$(awk -F'=' '/^
↪DISTRIB_DESCRIPTION=/ {print $2}' /etc/lsb-release-crunchbang)
                                elif [[ -f /etc/siduction-version ]]; then
                                        distro="Siduction"
```

(continues on next page)

(continued from previous page)

```
                                                distro_release="(Debian Sid)"
                                                distro_codename=""
                                        elif [[ -f /usr/bin/pveversion ]]; then
                                                distro="Proxmox VE"
                                                distro_more="$(/usr/bin/pveversion |grep -oP 'pve-manager\/\K\d+\.\d+')"
                                        elif [[ -f /etc/os-release ]]; then
                                                if [[ "$(cat /etc/os-release)" =~ "Raspbian" ]]; then
                                                        distro="Raspbian"
                                                        distro_release=$(awk -F'=' '/^PRETTY_NAME=/ {print $2}' /etc/os-release)
                                                fi
                                                if [[ "$(cat /etc/os-release)" =~ "BlankOn" ]]; then
                                                        distro="BlankOn"
                                                        distro_release=$(awk -F'=' '/^PRETTY_NAME=/ {print $2}' /etc/os-release)
                                                else
                                                        distro="Debian"
                                                fi
                                        else
                                                distro="Debian"
                                        fi
                                        ;;
                                "elementary"|"elementary OS")
                                        distro="elementary OS"
                                        ;;
                                "EvolveOS")
                                        distro="Evolve OS"
                                        ;;
                                "KaOS"|"kaos")
                                        distro="KaOS"
                                        ;;
                                "frugalware")
                                        distro="Frugalware"
                                        distro_codename=null
                                        distro_release=null
                                        ;;
                                "Fuduntu")
                                        distro="Fuduntu"
                                        distro_codename=null
                                        ;;
                                "Fux")
                                        distro="Fux"
                                        distro_codename=null
                                        ;;
                                "Gentoo")
                                        if [[ "$(lsb_release -sd)" =~ "Funtoo" ]];then
                                                distro="Funtoo"
                                        else
                                                distro="Gentoo"
                                        fi
                                        . /etc/portage/make.conf #detecting release stable/testing/experimental
                                        case $ACCEPT_KEYWORDS in
```

(continues on next page)

(continued from previous page)

```
                                                [a-z]*) distro_release=stable       ;;
                                                ~*)     distro_release=testing      ;;
                                                '**')   distro_release=experimental ;;
↪ #experimental usually includes git-versions.
                                        esac
                                        ;;
                                "Hyperbola GNU/Linux-libre"|"Hyperbola")
                                        distro="Hyperbola GNU/Linux-libre"
                                        distro_codename="n/a"
                                        distro_release="n/a"
                                        ;;
                                "LinuxDeepin")
                                        distro="LinuxDeepin"
                                        distro_codename=null
                                        ;;
                                "Kali"|"Debian Kali Linux")
                                        distro="Kali Linux"
                                        if [[ "${distro_codename}" =~ "kali-rolling"␣
↪]]; then
                                                distro_codename="n/a"
                                                distro_release="n/a"
                                        fi
                                        ;;
                                "Lunar Linux"|"lunar")
                                        distro="Lunar Linux"
                                        ;;
                                "MandrivaLinux")
                                        distro="Mandriva"
                                        case "${distro_codename}" in
                                                "turtle"|"Henry_Farman"|"Farman"|
↪"Adelie"|"pauillac")
                                                        distro="Mandriva-${distro_
↪release}"
                                                        distro_codename=null
                                                        ;;
                                        esac
                                        ;;
                                "ManjaroLinux")
                                        distro="Manjaro"
                                        ;;
                                "Mer")
                                        distro="Mer"
                                        if [[ -f /etc/os-release ]]; then
                                                if grep -q 'SailfishOS' /etc/os-
↪release; then
                                                        distro="SailfishOS"
                                                        distro_codename="$(grep
↪'VERSION=' /etc/os-release | cut -d '(' -f2 | cut -d ')' -f1)"
                                                        distro_release="$(awk -F'=' '/
↪^VERSION=/ {print $2}' /etc/os-release)"
                                                fi
                                        fi
                                        ;;
                                "neon"|"KDE neon")
                                        distro="KDE neon"
                                        distro_codename="n/a"
                                        distro_release="n/a"
```

(continues on next page)

(continued from previous page)

```
                                        if [[ -f /etc/issue ]]; then
                                                if grep -q "^KDE neon" /etc/issue ;␣
↪then
                                                        distro_release="$(grep '^KDE␣
↪neon' /etc/issue | cut -d ' ' -f3)"
                                                fi
                                        fi
                                        ;;
                                "Ol"|"ol"|"Oracle Linux")
                                        distro="Oracle Linux"
                                        [ -f /etc/oracle-release ] && distro_release="
↪$(sed 's/Oracle Linux //' /etc/oracle-release)"
                                        ;;
                                "LinuxMint")
                                        distro="Mint"
                                        if [[ "${distro_codename}" == "debian" ]];␣
↪then
                                                distro="LMDE"
                                                distro_codename="n/a"
                                                distro_release="n/a"
                                        fi
                                        ;;
                                "openSUSE"|"openSUSE project"|"SUSE LINUX")
                                        distro="openSUSE"
                                        if [ -f /etc/os-release ]; then
                                                if [[ "$(cat /etc/os-release)" =~
↪"SUSE Linux Enterprise" ]]; then
                                                        distro="SUSE Linux Enterprise"
                                                        distro_codename="n/a"
                                                        distro_release=$(awk -F'=' '/^
↪VERSION_ID=/ {print $2}' /etc/os-release | tr -d '"')
                                                fi
                                        fi
                                        if [[ "${distro_codename}" == "Tumbleweed" ]];
↪ then
                                                distro_release="n/a"
                                        fi
                                        ;;
                                "Parabola GNU/Linux-libre"|"Parabola")
                                        distro="Parabola GNU/Linux-libre"
                                        distro_codename="n/a"
                                        distro_release="n/a"
                                        ;;
                                "Parrot"|"Parrot Security")
                                        distro="Parrot Security"
                                        ;;
                                "PCLinuxOS")
                                        distro="PCLinuxOS"
                                        distro_codename="n/a"
                                        distro_release="n/a"
                                        ;;
                                "Peppermint")
                                        distro="Peppermint"
                                        distro_codename=null
                                        ;;
                                "rhel")
                                        distro="Redhat"
```

(continues on next page)

(continued from previous page)

```
                                        distro_release="$(redhat_release_version)"
                                        distro_codename="$(redhat_codename_version)"
                                        ;;
                                "RosaDesktopFresh")
                                        distro="ROSA"
                                        distro_release=$(grep 'VERSION=' /etc/os-
↪release | cut -d ' ' -f3 | cut -d "\"" -f1)
                                        distro_codename=$(grep 'PRETTY_NAME=' /etc/os-
↪release | cut -d ' ' -f4,4)
                                        ;;
                                "SailfishOS")
                                        distro="SailfishOS"
                                        if [[ -f /etc/os-release ]]; then
                                                distro_codename="$(grep 'VERSION=' /
↪etc/os-release | cut -d '(' -f2 | cut -d ')' -f1)"
                                                distro_release="$(awk -F'=' '/^
↪VERSION=/ {print $2}' /etc/os-release)"
                                        fi
                                        ;;
                                "Sparky"|"SparkyLinux")
                                        distro="SparkyLinux"
                                        ;;
                                "Viperr")
                                        distro="Viperr"
                                        distro_codename=null
                                        ;;
                                "Void"|"VoidLinux")
                                        distro="Void Linux"
                                        distro_codename=""
                                        distro_release=""
                                        ;;
                                "Zorin")
                                        distro="Zorin OS"
                                        distro_codename=""
                                        ;;
                                *)
                                        if [ "x$(printf "${distro_detect}" | od -t x1␣
↪| sed -e 's/^\w*\ *//' | tr '\n' ' ' | grep 'eb b6 89 ec 9d 80 eb b3 84 ')" != "x"␣
↪]; then
                                                distro="Red Star OS"
                                                distro_codename="n/a"
                                                distro_release=$(printf "${distro_
↪release}" | grep -o '[0-9.]' | tr -d '\n')
                                        fi
                                        ;;
                        esac
                        if [[ "${distro_detect}" =~ "RedHatEnterprise" ]]; then
                                distro="Redhat"
                                distro_release="$(redhat_release_version)"
                                distro_codename="$(redhat_codename_version)"
                        fi
                        if [[ "${distro_detect}" =~ "SUSELinuxEnterprise" ]]; then␣
↪distro="SUSE Linux Enterprise"; fi
                        if [[ -n ${distro_release} && ${distro_release} != "n/a" ]];␣
↪then distro_more="$distro_release"; fi
                        if [[ -n ${distro_codename} && ${distro_codename} != "n/a" ]];
↪ then distro_more="$distro_more $distro_codename"; fi
```

(continues on next page)

(continued from previous page)

```
                fi

                # Existing File Check
                if [ "$distro" == "Unknown" ]; then
                        if [ $(uname -o 2>/dev/null) ]; then
                                os="$(uname -o)"
                                case "$os" in
                                        "Cygwin"|"FreeBSD"|"OpenBSD"|"NetBSD")
                                                distro="$os"
                                                fake_distro="${distro}"
                                        ;;
                                        "DragonFly")
                                                distro="DragonFlyBSD"
                                                fake_distro="${distro}"
                                        ;;
                                        "Msys")
                                                distro="Msys"
                                                fake_distro="${distro}"
                                                distro_more="${distro} $(uname -r |
↪head -c 1)"
                                        ;;
                                        "Haiku")
                                                distro="Haiku"
                                                distro_more="$(uname -v | tr ' ' '\n'
↪| grep 'hrev')"
                                        ;;
                                        "GNU/Linux")
                                                if type -p crux >/dev/null 2>&1; then
                                                        distro="CRUX"
                                                        distro_more="$(crux | awk '
↪{print $3}')"
                                                fi
                                                if type -p nixos-version >/dev/null 2>
↪&1; then
                                                        distro="NixOS"
                                                        distro_more="$(nixos-version)"
                                                fi
                                                if type -p sorcery >/dev/null 2>&1;
↪then
                                                        distro="SMGL"
                                                fi
                                                if (type -p guix && type -p herd) >/
↪dev/null 2>&1; then
                                                        distro="GuixSD"
                                                fi
                                        ;;
                                esac
                        fi
                        if [[ "${distro}" == "Cygwin" || "${distro}" == "Msys" ]];
↪then
                                # https://msdn.microsoft.com/en-us/library/ms724832
↪%28VS.85%29.aspx
                                if [ "$(wmic os get version | grep -o '^\(6\.
↪[23]\|10\)')" ]; then
                                        fake_distro="Windows - Modern"
                                fi
                        fi
```

(continues on next page)

(continued from previous page)

```
                        if [[ "${distro}" == "Unknown" ]]; then
                                if [ -f /etc/os-release ]; then
                                        os_release="/etc/os-release";
                                elif [ -f /usr/lib/os-release ]; then
                                        os_release="/usr/lib/os-release";
                                fi
                                if [[ -n ${os_release} ]]; then
                                        distrib_id=$(<${os_release});
                                        for l in $(echo $distrib_id); do
                                                if [[ ${l} =~ ^ID= ]]; then
                                                        distrib_id=${l//*=}
                                                        distrib_id=${distrib_id//\"/}
                                                        break 1
                                                fi
                                        done
                                        if [[ -n ${distrib_id} ]]; then
                                                if [[ -n ${BASH_VERSINFO} && ${BASH_
↪VERSINFO} -ge 4 ]]; then
                                                        distrib_id=$(for i in $
↪{distrib_id}; do echo -n "${i^} "; done)
                                                        distro=${distrib_id% }
                                                        unset distrib_id
                                                else
                                                        distrib_id=$(for i in $
↪{distrib_id}; do FIRST_LETTER=$(echo -n "${i:0:1}" | tr "[:lower:]" "[:upper:]");␣
↪echo -n "${FIRST_LETTER}${i:1} "; done)
                                                        distro=${distrib_id% }
                                                        unset distrib_id
                                                fi
                                        fi

                                        # Hotfixes
                                        [[ "${distro}" == "void" ]] && distro="Void␣
↪Linux"
                                        [[ "${distro}" == "evolveos" ]] && distro=
↪"Evolve OS"
                                        [[ "${distro}" == "antergos" ]] && distro=
↪"Antergos"
                                        [[ "${distro}" == "logos" ]] && distro="Logos"
                                        [[ "${distro}" == "Arch" || "${distro}" ==
↪"Archarm" || "${distro}" == "archarm" ]] && distro="Arch Linux"
                                        [[ "${distro}" == "elementary" ]] && distro=
↪"elementary OS"
                                        [[ "${distro}" == "Fedora" && -d /etc/qubes-
↪rpc ]] && distro="qubes" # Inner VM
                                        [[ "${distro}" == "Ol" || "${distro}" == "ol"␣
↪]] && distro="Oracle Linux"
                                        if [[ "${distro}" == "Oracle Linux" ]] && [ -
↪f /etc/oracle-release ]; then
                                                distro_more="$(sed 's/Oracle Linux //
↪' /etc/oracle-release)"
                                        fi
                                        [[ "${distro}" == "rhel" ]] && distro="Redhat"
                                        [[ "${distro}" == "Neon" ]] && distro="KDE␣
↪neon"
                                        [[ "${distro}" == "SLED" || "${distro}" ==
↪"sled" || "${distro}" == "SLES" || "${distro}" == "sles" ]] && distro="SUSE Linux␣
↪Enterprise"
```

(continues on next page)

(continued from previous page)

```
                                        if [[ "${distro}" == "SUSE Linux Enterprise" ]] && [ -f /etc/os-release ]; then
                                                distro_more="$(awk -F'=' '/^VERSION_ID=/ {print $2}' /etc/os-release | tr -d '"')"
                                        fi
                                        if [[ "${distro}" == "Debian" ]] && [ -f /usr/bin/pveversion ]; then
                                                distro="Proxmox VE"
                                                distro_more="$(/usr/bin/pveversion | grep -oP 'pve-manager\/\K\d+\.\d+')"
                                        fi
                                fi
                        fi

                        if [[ "${distro}" == "Unknown" ]]; then
                                if [[ "${OSTYPE}" == "linux-gnu" || "${OSTYPE}" == "linux" ]]; then
                                        if [ -f /etc/lsb-release ]; then
                                                LSB_RELEASE=$(</etc/lsb-release)
                                                distro=$(echo ${LSB_RELEASE} | awk 'BEGIN {
                                                        distro = "Unknown"
                                                }
                                                {
                                                        if ($0 ~ /[Uu][Bb][Uu][Nn][Tt][Uu]/) {
                                                                distro = "Ubuntu"
                                                                exit
                                                        }
                                                        else if ($0 ~ /[Mm][Ii][Nn][Tt]/ && $0 ~ /[Dd][Ee][Bb][Ii][Aa][Nn]/) {
                                                                distro = "LMDE"
                                                                exit
                                                        }
                                                        else if ($0 ~ /[Mm][Ii][Nn][Tt]/) {
                                                                distro = "Mint"
                                                                exit
                                                        }
                                                } END {
                                                        print distro
                                                }')
                                        fi
                                fi
                        fi

                        if [[ "${distro}" == "Unknown" ]] && [[ "${OSTYPE}" == "linux-gnu" || "${OSTYPE}" == "linux" || "${OSTYPE}" == "gnu" ]]; then
                                for di in arch chakra crunchbang-lsb evolveos exherbo fedora \
                                                frugalware fux gentoo kogaion mageia obarun oracle \
                                                pardus pclinuxos redhat redstar rosa SuSe; do
                                        if [ -f /etc/$di-release ]; then distro=$di && break; fi
                                done
```

(continues on next page)

(continued from previous page)

```
                                        if [[ "${distro}" == "crunchbang-lsb" ]]; then
                                                distro == "Crunchbang"
                                        elif [[ "${distro}" == "gentoo" ]]; then
                                                grep -q "Funtoo" /etc/gentoo-release &
↪& distro="Funtoo"
                                        elif [[ "${distro}" == "mandrake" ]] || [[ "$
↪{distro}" == "mandriva" ]]; then
                                                grep -q "PCLinuxOS" /etc/${distro}-
↪release && distro="PCLinuxOS"
                                        elif [[ "${distro}" == "fedora" ]]; then
                                                grep -q "Korora" /etc/fedora-release &
↪& distro="Korora"
                                                grep -q "BLAG" /etc/fedora-release &&␣
↪distro="BLAG" && distro_more="$(head -n1 /etc/fedora-release)"
                                        elif [[ "${distro}" == "oracle" ]]; then
                                                distro_more="$(sed 's/Oracle Linux //
↪' /etc/oracle-release)"
                                        elif [[ "${distro}" == "SuSe" ]]; then
                                                distro="openSUSE"
                                                if [ -f /etc/os-release ]; then
                                                        if [[ "$(cat /etc/os-release)
↪" =~ "SUSE Linux Enterprise" ]]; then
                                                                distro="SUSE Linux␣
↪Enterprise"
                                                                distro_more=$(awk -F'=
↪' '/^VERSION_ID=/ {print $2}' /etc/os-release | tr -d '"')
                                                        fi
                                                fi
                                                if [[ "${distro_more}" =~ "Tumbleweed
↪" ]]; then
                                                        distro_more="Tumbleweed"
                                                fi
                                        elif [[ "${distro}" == "redstar" ]]; then
                                                distro_more=$(grep -o '[0-9.]' /etc/
↪redstar-release | tr -d '\n')
                                        elif [[ "${distro}" == "redhat" ]]; then
                                                grep -q "CentOS" /etc/redhat-release &
↪& distro="CentOS"
                                                grep -q "PCLinuxOS" /etc/redhat-
↪release && distro="PCLinuxOS"
                                                if [ "x$(od -t x1 /etc/redhat-release␣
↪| sed -e 's/^\w*\ */' | tr '\n' ' ' | grep 'eb b6 89 ec 9d 80 eb b3 84 ')" != "x"␣
↪]; then
                                                        distro="Red Star OS"
                                                        distro_more=$(grep -o '[0-9.]
↪' /etc/redhat-release | tr -d '\n')
                                                fi
                                                distro_release="$(redhat_release_
↪version)"
                                                distro_codename="$(redhat_codename_
↪version)"
                                        fi
                                fi

                                if [[ "${distro}" == "Unknown" ]]; then
                                        if [[ "${OSTYPE}" == "linux-gnu" || "${OSTYPE}" ==
↪"linux" || "${OSTYPE}" == "gnu" ]]; then
```

(continues on next page)

(continued from previous page)

```
                                    if [ -f /etc/debian_version ]; then
                                        if [ -f /etc/issue ]; then
                                            if grep -q "gNewSense" /etc/
↪issue ; then
                                                distro="gNewSense"
                                            elif grep -q "^KDE neon" /etc/
↪issue ; then
                                                distro="KDE neon"
                                                distro_more="$(cut -d
↪' ' -f3 /etc/issue)"
                                            else
                                                distro="Debian"
                                            fi
                                        fi
                                        if grep -q "Kali" /etc/debian_version␣
↪; then
                                            distro="Kali Linux"
                                        fi
                                    elif [ -f /etc/NIXOS ]; then distro="NixOS"
                                    elif [ -f /etc/dragora-version ]; then
                                        distro="Dragora"
                                        distro_more="$(cut -d, -f1 /etc/
↪dragora-version)"
                                    elif [ -f /etc/slackware-version ]; then␣
↪distro="Slackware"
                                    elif [ -f /usr/share/doc/tc/release.txt ];␣
↪then
                                        distro="TinyCore"
                                        distro_more="$(cat /usr/share/doc/tc/
↪release.txt)"
                                    elif [ -f /etc/sabayon-edition ]; then distro=
↪"Sabayon"
                                    fi
                                else
                                    if [[ -x /usr/bin/sw_vers ]] && /usr/bin/sw_
↪vers | grep -i "Mac OS X" >/dev/null; then
                                        distro="Mac OS X"
                                    elif [[ -f /var/run/dmesg.boot ]]; then
                                        distro=$(awk 'BEGIN {
                                            distro = "Unknown"
                                        }
                                        {
                                            if ($0 ~ /DragonFly/) {
                                                distro = "DragonFlyBSD
↪"
                                                exit
                                            }
                                            else if ($0 ~ /FreeBSD/) {
                                                distro = "FreeBSD"
                                                exit
                                            }
                                            else if ($0 ~ /NetBSD/) {
                                                distro = "NetBSD"
                                                exit
                                            }
                                            else if ($0 ~ /OpenBSD/) {
                                                distro = "OpenBSD"
```

(continues on next page)

(continued from previous page)

```
                                                                exit
                                                        }
                                                } END {
                                                        print distro
                                                }' /var/run/dmesg.boot)
                                        fi
                                fi
                        fi

                        if [[ "${distro}" == "Unknown" ]] && [[ "${OSTYPE}" == "linux-gnu" || "${OSTYPE}" == "linux" || "${OSTYPE}" == "gnu" ]]; then
                                if [[ -f /etc/issue ]]; then
                                        distro=$(awk 'BEGIN {
                                                distro = "Unknown"
                                        }
                                        {
                                                if ($0 ~ /"Hyperbola GNU\/Linux-libre"/) {
                                                        distro = "Hyperbola GNU/Linux-libre"
                                                        exit
                                                }
                                                else if ($0 ~ /"LinuxDeepin"/) {
                                                        distro = "LinuxDeepin"
                                                        exit
                                                }
                                                else if ($0 ~ /"Obarun"/) {
                                                        distro = "Obarun"
                                                        exit
                                                }
                                                else if ($0 ~ /"Parabola GNU\/Linux-libre"/) {
                                                        distro = "Parabola GNU/Linux-libre"
                                                        exit
                                                }
                                                else if ($0 ~ /"Solus"/) {
                                                        distro = "Solus"
                                                        exit
                                                }
                                                else if ($0 ~ /"ALDOS"/) {
                                                        distro = "ALDOS"
                                                        exit
                                                }
                                        } END {
                                                print distro
                                        }' /etc/issue)
                                fi
                        fi

                        if [[ "${distro}" == "Unknown" ]] && [[ "${OSTYPE}" == "linux-gnu" || "${OSTYPE}" == "linux" || "${OSTYPE}" == "gnu" ]]; then
                                if [[ -f /etc/system-release ]]; then
                                        if grep -q "Scientific Linux" /etc/system-release; then
                                                distro="Scientific Linux"
                                        elif grep -q "Oracle Linux" /etc/system-release; then
```

(continues on next page)

(continued from previous page)

```
                                        distro="Oracle Linux"
                                fi
                        elif [[ -f /etc/lsb-release ]]; then
                                if grep -q "CHROMEOS_RELEASE_NAME" /etc/lsb-release; then
                                        distro="$(awk -F'=' '/^CHROMEOS_RELEASE_NAME=/ {print $2}' /etc/lsb-release)"
                                        distro_more="$(awk -F'=' '/^CHROMEOS_RELEASE_VERSION=/ {print $2}' /etc/lsb-release)"
                                fi
                        fi
                fi
        fi

        if [[ -n ${distro_more} ]]; then
                distro_more="${distro} ${distro_more}"
        fi

        if [[ "${distro}" != "Haiku" ]]; then
                if [[ ${BASH_VERSINFO[0]} -ge 4 ]]; then
                        if [[ ${BASH_VERSINFO[0]} -eq 4 && ${BASH_VERSINFO[1]} -gt 1 ]] || [[ ${BASH_VERSINFO[0]} -gt 4 ]]; then
                                distro=${distro,,}
                        else
                                distro="$(tr '[:upper:]' '[:lower:]' <<< ${distro})"
                        fi
                else
                        distro="$(tr '[:upper:]' '[:lower:]' <<< ${distro})"
                fi
        fi

        case $distro in
                aldos) distro="ALDOS";;
                alpine) distro="Alpine Linux" ;;
                amzn|amazon|amazon*linux)
                        distro="AmazonLinux"
                        if [ "$(grep VERSION_ID /etc/os-release | awk -F '=' '{print $2}')" = '"2"' ]; then
                                distro_release="2"
                        elif [ "${amazonlinux_release_version}" != "2" ]; then
                                distro_release="1"
                        else
                                distro_release="unknown"
                        fi
                        ;;
                antergos) distro="Antergos" ;;
                arch*linux*old) distro="Arch Linux - Old" ;;
                arch|arch*linux) distro="Arch Linux" ;;
                artix|artix*linux) distro="Artix Linux" ;;
                blackpantheros|black*panther*) distro="blackPanther OS" ;;
                blag) distro="BLAG" ;;
                bunsenlabs) distro="BunsenLabs" ;;
                centos)
                        distro="CentOS"
                        distro_release="$(redhat_release_version)"
                        distro_codename="$(redhat_codename_version)"
```

(continues on next page)

(continued from previous page)

```
                        ;;
                chakra) distro="Chakra" ;;
                chapeau) distro="Chapeau" ;;
                chrome*|chromium*) distro="Chrome OS" ;;
                crunchbang) distro="CrunchBang" ;;
                crux) distro="CRUX" ;;
                cygwin) distro="Cygwin" ;;
                debian) distro="Debian" ;;
                devuan) distro="Devuan" ;;
                deepin) distro="Deepin" ;;
                desaos) distro="DesaOS" ;;
                dragonflybsd) distro="DragonFlyBSD" ;;
                dragora) distro="Dragora" ;;
                elementary|'elementary os') distro="elementary OS";;
                evolveos) distro="Evolve OS" ;;
                exherbo|exherbo*linux) distro="Exherbo" ;;
                fedora)
                        distro="Fedora"
                        distro_release="$(fedora_release_version)"
                        distro_codename="$(fedora_codename_version)"
                        ;;
                freebsd) distro="FreeBSD" ;;
                freebsd*old) distro="FreeBSD - Old" ;;
                frugalware) distro="Frugalware" ;;
                fuduntu) distro="Fuduntu" ;;
                funtoo) distro="Funtoo" ;;
                fux) distro="Fux" ;;
                gentoo) distro="Gentoo" ;;
                gnewsense) distro="gNewSense" ;;
                guixsd) distro="GuixSD" ;;
                haiku) distro="Haiku" ;;
                hyperbolagnu|hyperbolagnu/linux-libre|'hyperbola gnu/linux-libre
↪'|hyperbola) distro="Hyperbola GNU/Linux-libre" ;;
                kali*linux) distro="Kali Linux" ;;
                kaos) distro="KaOS";;
                kde*neon|neon) distro="KDE neon" ;;
                kogaion) distro="Kogaion" ;;
                korora) distro="Korora" ;;
                linuxdeepin) distro="LinuxDeepin" ;;
                lmde) distro="LMDE" ;;
                logos) distro="Logos" ;;
                lunar|lunar*linux) distro="Lunar Linux";;
                mac*os*x|os*x) distro="Mac OS X" ;;
                manjaro) distro="Manjaro" ;;
                mageia) distro="Mageia" ;;
                mandrake) distro="Mandrake" ;;
                mandriva) distro="Mandriva" ;;
                mer) distro="Mer" ;;
                mint|linux*mint) distro="Mint" ;;
                msys|msys2) distro="Msys" ;;
                netbsd) distro="NetBSD" ;;
                netrunner) distro="Netrunner" ;;
                nix|nix*os) distro="NixOS" ;;
                obarun) distro="Obarun" ;;
                obrevenge) distro="OBRevenge" ;;
                ol|oracle*linux) distro="Oracle Linux" ;;
                openbsd) distro="OpenBSD" ;;
```

(continues on next page)

(continued from previous page)

```
                opensuse) distro="openSUSE" ;;
                parabolagnu|parabolagnu/linux-libre|'parabola gnu/linux-libre
↪'|parabola) distro="Parabola GNU/Linux-libre" ;;
                pardus) distro="Pardus" ;;
                parrot|parrot*security) distro="Parrot Security" ;;
                pclinuxos|pclos) distro="PCLinuxOS" ;;
                peppermint) distro="Peppermint" ;;
                proxmox|proxmox*ve) distro="Proxmox VE" ;;
                qubes) distro="Qubes OS" ;;
                raspbian) distro="Raspbian" ;;
                red*hat*|rhel)
                        distro="Redhat"
                        distro_release="$(redhat_release_version)"
                        distro_codename="$(redhat_codename_version)"
                        ;;
                rosa) distro="ROSA" ;;
                red*star|red*star*os) distro="Red Star OS" ;;
                sabayon) distro="Sabayon" ;;
                sailfish|sailfish*os) distro="SailfishOS" ;;
                siduction) distro="Siduction" ;;
                slackware) distro="Slackware" ;;
                smgl|source*mage|source*mage*gnu*linux) distro="Source Mage GNU/Linux
↪" ;;
                solus) distro="Solus" ;;
                sparky|sparky*linux) distro="SparkyLinux" ;;
                steam|steam*os) distro="SteamOS" ;;
                suse*linux*enterprise) distro="SUSE Linux Enterprise" ;;
                swagarch) distro="SwagArch" ;;
                tinycore|tinycore*linux) distro="TinyCore" ;;
                trisquel) distro="Trisquel";;
                grombyangos) distro="GrombyangOS" ;;
                ubuntu)
                        distro="Ubuntu"
                        if grep -q 'Microsoft' /proc/version 2>/dev/null || \
                           grep -q 'Microsoft' /proc/sys/kernel/osrelease 2>/dev/null
                        then
                                uow=$(echo -e "$(getColor 'yellow') [Ubuntu on␣
↪Windows 10]")
                        fi
                        ;;
                viperr) distro="Viperr" ;;
                void*linux) distro="Void Linux" ;;
                zorin*) distro="Zorin OS" ;;
        esac
        if [ "$format" = "json" ]; then
                echo "{\"Major\": \"${distro}\", \"Release\": \"${distro_release}\", \
↪"Codename\": \"${distro_codename}\"}"
        else
                echo "${distro} ${distro_release} ${distro_codename}"
        fi
}

# call main
detectdistro
```

Back to *Bash Module Index*

Table Of Contents

24.7 std_functions.sh

Back to *Bash Module Index*

```
#!/usr/bin/env bash

#-------------------------------------------------------------------------------
#
#   Note:  to be used with dependent modules
#
#       - colors.sh
#       - exitcodes.sh
#
#       Dependencies must be sourced from the same calling script
#       as this std_functions.sh
#
#   Global Variables provided by the Caller:
#       - LOG_FILE      # std_logger writes to this file
#       - QUIET         # Value = "true" to supress stdout from these reference functions
#
#-------------------------------------------------------------------------------

 # pkg reported in logs will be the basename of the caller
pkg=$(basename $0 2>/dev/null)
pkg_root="$(echo $pkg | awk -F '.' '{print $1}')"       # pkg without file extention
pkg_path=$(cd $(dirname $0 2>/dev/null); pwd -P)
host=$(hostname)
system=$(uname)

# this file
LIB_VERSION="2.9.8"

if [ ! $pkg ] || [ ! $pkg_path ]; then
    echo -e "\n[std_functions.sh]: pkg and pkg_path errors - both are null"
fi


function array2json(){
    ## converts associative array to single-level (no nested keys) json file output ##
    #
    #   Caller syntax:
    #       $ array2json config_dict $config_path/configuration_file
    #
    #   where:
    #       $ declare -A config_dict        # config_dict is assoc array, declared in main script
    #
    local -n array_dict=$1      # local assoc array must use -n opt
```

(continues on next page)

(continued from previous page)

```
    local output_file=$2        # location
    local ct                    # counter
    local max_keys              # num keys in array
    #
    echo -e "{" > $output_file
    ct=1
    max_keys=${#array_dict[@]}
    for key in "${!array_dict[@]}"; do
        if [ $ct == $max_keys ]; then
            # last key, no comma
            echo "\"${key}\": \"${array_dict[${key}]}\"" | indent04 >> $output_file
        else
            echo "\"${key}\": \"${array_dict[${key}]}\"," | indent04 >> $output_file
        fi
        ct=$(( $ct + 1 ))
    done
    echo -e "}" >> $output_file
    #
    # <-- end function array2json -->
}


function authenticated(){
    ## validates authentication using iam user or role ##
    local profilename="$1"
    local response
    local awscli=$(which aws)
    #
    response=$($awscli sts get-caller-identity --profile $profilename 2>&1)
    if [ "$(echo $response | grep Invalid)" ]; then
        std_message "The IAM profile provided ($profilename) failed to authenticate to AWS. Exit (Code $E_AUTH)" "AUTH"
        return 1
    elif [ "$(echo $response | grep found)" ]; then
        std_message "The IAM user or role ($profilename) cannot be found in your local awscli config. Exit (Code $E_BADARG)" "AUTH"
        return 1
    elif [ "$(echo $response | grep Expired)" ]; then
        std_message "The sts temporary credentials for the role provided ($profilename) have expired. Exit (Code $E_AUTH)" "INFO"
        return 1
    else
        return 0
    fi
}


function binary_depcheck(){
    ## validate binary dependencies installed
    local check_list=( "$@" )
    local msg
    #
    for prog in "${check_list[@]}"; do
        if ! type "$prog" > /dev/null 2>&1; then
            msg="$prog is required and not found in the PATH. Aborting (code $E_DEPENDENCY)"
            std_error_exit "$msg" $E_DEPENDENCY
```

(continues on next page)

(continued from previous page)

```
        fi
    done
    #
    # <<-- end function binary_depcheck -->>
}


function convert_time(){
    # time format conversion (http://stackoverflow.com/users/1030675/choroba)
    num=$1
    min=0
    hour=0
    day=0
    if((num>59));then
        ((sec=num%60))
        ((num=num/60))
        if((num>59));then
            ((min=num%60))
            ((num=num/60))
            if((num>23));then
                ((hour=num%24))
                ((day=num/24))
            else
                ((hour=num))
            fi
        else
            ((min=num))
        fi
    else
        ((sec=num))
    fi
    echo "$day"d,"$hour"h,"$min"m
    #
    # <-- end function convert_time -->
    #
}


function convert_time_months(){
    # time format conversion (http://stackoverflow.com/users/1030675/choroba)
    num=$1
    min=0
    hour=0
    day=0
    mo=0
    if((num>59));then
        ((sec=num%60))
        ((num=num/60))
        if((num>59));then
            ((min=num%60))
            ((num=num/60))
            if((num>23));then
                ((hour=num%24))
                ((day=num/24))
                ((num=num/24))
                if((num>30)); then
                  ((day=num%31))
```

(continues on next page)

(continued from previous page)

```
                    ((mo=num/30))
                else
                    ((day=num))
                fi
            else
                ((hour=num))
            fi
        else
            ((min=num))
        fi
    else
        ((sec=num))
    fi
    if (( $mo > 0 )); then
        echo -e "$mo"m,"$day"d
    else
        echo -e "$day"d,"$hour"h,"$min"m
    fi
    #
    # <-- end function convert_time -->
    #
}


function delay_spinner(){
    ##
    ##  Usage:
    ##
    ##      $ long-running-command  &
    ##      $ delay_spinner "  Please wait msg..."
    ##
    ##  Spinner exists when long-running-command completes
    ##
    local PROGRESSTXT
    if [ ! "$1" ]; then
        PROGRESSTXT="  Please wait..."
    else
        PROGRESSTXT="$1"
    fi
    # visual progress marker function
    # http://stackoverflow.com/users/2869509/wizurd
    # vars
    local pid=$!
    local delay=0.1
    local spinstr='|/-\'
    echo -e "\n\n"
    while [ "$(ps a | awk '{print $1}' | grep $pid)" ]; do
        local temp=${spinstr#?}
        printf "\r$PROGRESSTXT[%c]  " "$spinstr"
        local spinstr=$temp${spinstr%"$temp"}
        sleep $delay
        printf "\b\b\b\b\b\b"
    done
    printf -- '\n\n'
    #
    # <-- end function ec2cli_spinner -->
    #
```

(continues on next page)

(continued from previous page)

```
}


function environment_info(){
    local prefix=$1
    local dep=$2
    local log_file="$3"
    local version_info
    local awscli_ver
    local boto_ver
    local python_ver
    #
    version_info=$(aws --version 2>&1)
    awscli_ver=$(echo $version_info | awk '{print $1}')
    boto_ver=$(echo $version_info | awk '{print $4}')
    python_ver=$(echo $version_info | awk '{print $2}')
    #
    if [[ $dep == "aws" ]]; then
        std_logger "awscli version detected: $awscli_ver" $prefix $log_file
        std_logger "Python runtime detected: $python_ver" $prefix $log_file
        std_logger "Kernel detected: $(echo $version_info | awk '{print $3}')"
↪$prefix $log_file
        std_logger "boto library detected: $boto_ver" $prefix $log_file

    elif [[ $dep == "awscli" ]]; then
        std_message "awscli version detected: ${accent}${BOLD}$awscli_ver${UNBOLD}$
↪{reset}" $prefix $log_file | indent04
        std_message "boto library detected: ${accent}${BOLD}$boto_ver${UNBOLD}${reset}
↪" $prefix $log_file | indent04
        std_message "Python runtime detected: ${accent}${BOLD}$python_ver${UNBOLD}$
↪{reset}" $prefix $log_file | indent04

    elif [[ $dep == "os" ]]; then
        std_message "Kernel detected: ${title}$(echo $version_info | awk '{print $3}')
↪${reset}" $prefix $log_file | indent04

    elif [[ $dep == "jq" ]]; then
        version_info=$(jq --version 2>&1)
        std_message "JSON parser detected: ${title}$(echo $version_info)${reset}"
↪$prefix $log_file | indent04

    else
        std_logger "Detected: $($prog --version | head -1)" $prefix $log_file
    fi
    #
    # <<-- end function environment_info -->>
}


function is_installed(){
    ##
    ## validate if binary previously installed  ##
    ##
    local binary="$1"
    local location=$(which $binary 2>/dev/null)

    if [ $location ]; then
```

(continues on next page)

(continued from previous page)

```
262
263         std_message "$binary is installed:  $location" "INFO" $LOG_FILE
264         return 0
265
266     else
267
268         return 1
269
270     fi
271     #
272     #<-- end function is_installed -->
273 }
274
275
276 function is_float(){
277     ##
278     ## Checks type for floating point number
279     ## see is_number integer type checking
280     ##
281     local num="$1"
282     local regex='^[0-9]+[.][0-9]+?$'
283
284     if [[ $num =~ $regex ]] ; then
285
286         # int or float
287         return 0
288
289     fi
290
291     return 1        # not a floating point number
292     #
293     #<-- end function is_float -->
294 }
295
296
297 function is_int(){
298     ##
299     ## see is_float for decimal type checking ##
300     ##
301     local num="$1"
302     local regex='^[0-9]+$'
303
304     if [[ $num =~ $regex ]] ; then
305
306         # int or float
307         return 0
308
309     fi
310
311     return 1        # not an integer
312     #
313     #<-- end function is_int -->
314 }
315
316
317 function is_number(){
318     ##
```

(continues on next page)

(continued from previous page)

```
    ## type checking; any, int or decimal type ##
    ##
    local num="$1"
    local regex='^[0-9]+([.][0-9]+)?$'

    if [[ $num =~ $regex ]] ; then

        # int or float
        return 0

    fi

    return 1        # not a number (is alpha character)
    #
    #<-- end function is_number -->
}


function linux_distro(){
    ##
    ## determine linux os distribution ##
    ##
    local os_major
    local os_release
    local os_codename
    declare -a distro_info

    if [ "$(which lsb_release)" ]; then

        distro_info=( $(lsb_release -sirc) )

        if [[ ${#distro_detect[@]} -eq 3 ]]; then
            os_major=${distro_info[0]}
            os_release=${distro_info[1]}
            os_codename=${distro_info[2]}
        fi

    else

        ## AMAZON Linux ##
        if [ "$(grep -i amazon /etc/os-release  | head -n 1)" ]; then

            os_major="amazonlinux"
            if [ "$(grep VERSION_ID /etc/os-release | awk -F '=' '{print $2}')" = '"2"
↪' ]; then
                os_release="$(grep VERSION /etc/os-release | grep -v VERSION_ID | awk␣
↪-F '=' '{print $2}')"
                os_release=$(echo $os_release | cut -c 2-15 | rev | cut -c 2-15 | rev)
            elif [ "$(grep VERSION_ID /etc/os-release | awk -F '=' '{print $2}')" = '
↪"1"' ]; then
                os_release="$(grep VERSION /etc/os-release | grep -v VERSION_ID | awk␣
↪-F '=' '{print $2}')"
                os_release=$(echo $os_release | cut -c 2-15 | rev | cut -c 2-15 | rev)
            else os_release="unknown"; fi

        ## REDHAT Linux ##
        elif [ $(grep -i redhat /etc/os-release  | head -n 1) ]; then
```

(continues on next page)

(continued from previous page)

```
            os_major="redhat"
            os_release="future"

        ## UBUNTU, ubuntu variants ##
        elif [ "$(grep -i ubuntu /etc/os-release)" ]; then

            os_major="ubuntu"
            if [ "$(grep -i mint /etc/os-release | head -n1)" ]; then
                os_release="linuxmint"
            elif [ "$(grep -i ubuntu_codename /etc/os-release | awk -F '=' '{print $2}
↪')" ]; then
                os_release="$(grep -i ubuntu_codename /etc/os-release | awk -F '=' '
↪{print $2}')"
            else
                os_release="unknown"; fi

        ## distribution not determined ##
        else

            os_major="unknown"; os_release="unknown"

        fi

    fi

    # set distribution type in environment
    export OS_DISTRO="$os_major"
    std_logger "Operating system identified as Major Version: $os_major, Minor␣
↪Version: $os_release" "INFO" $LOG_FILE

    # return major, minor disto versions
    echo -e "$os_major $os_release $os_codename"
    #
    # <<--- end function linux_distro -->>
}


function pkg_info(){
    ##
    ##  displays information about this library module
    ##
    ##      - dependent module colors.sh is located always adjacent
    ##      - sourcing of dep modules must occur after local var to avoid overwrite
    ##        of variable values in this module
    ##
    local version=$LIB_VERSION
    source $pkg_path/colors.sh
    bd=$(echo -e ${bold})
    act=$(echo -e ${a_orange})
    rst=$(echo -e ${reset})

    # generate list of functions
    printf -- '%s\n' "$(declare -F | awk '{print $3}')" > /tmp/.functions
    sum=$(cat /tmp/.functions | wc -l)

    # construct, display help msg output
```

(continues on next page)

(continued from previous page)

```
    cat <<EOM
    ------------------------------------------------------------

    ${title}Bashtools Library${rst}: Standard Functions

    Module Name:         ${cyan}$pkg${rst}
    Module Version:      ${act}$version${rst}
    ------------------------------------------------------------

    Module Contains $sum Functions:

EOM
    # display list of function names in this module
    for l in $(cat /tmp/.functions); do
        printf -- '\t%s %s\n' "-" "$1"
    done
    printf -- '\n'
    rm /tmp/.functions
    #
    # <<-- end function pkg_info -->>
}


function print_header(){
    ##
    ## print formatted report header ##
    ##
    local title="$1"
    local width="$2"
    local reportfile="$3"
    #
    #if (( $(tput cols) > 480 )); then
    #    printf "%-10s %*s\n" $(echo -e ${frame}) "$(($width - 1))" '' | tr ' ' _ |␣
↪indent02 > $reportfile
    #else
        printf "%-10s %*s" $(echo -e ${frame}) "$(($width - 1))" '' | tr ' ' _ |␣
↪indent02 > $reportfile
    #fi
    echo -e "${bodytext}" >> $reportfile
    echo -ne ${title} >> $reportfile
    echo -e "${frame}" >> $reportfile
    printf '%*s' "$width" '' | tr ' ' _  | indent02 >> $reportfile
    echo -e "${bodytext}" >> $reportfile
    #
    # <<--- end function print_header -->>
}


function print_footer(){
    ##
    ## print formatted report footer ##
    ##
    local footer="$1"
    local width="$2"

    printf "%-10s %*s\n" $(echo -e ${frame}) "$(($width - 1))" '' | tr ' ' _ |␣
↪indent02
```

(continues on next page)

(continued from previous page)

```
    echo -e "${bodytext}"
    echo -ne $footer | indent20
    echo -e "${frame}"
    printf '%*s\n' "$width" '' | tr ' ' _ | indent02
    echo -e "${bodytext}"
    #
    # <<--- end function print_footer -->>
}


function print_separator(){
    ##
    ## prints single bar separator of width ##
    ##

    local width="$1"

    echo -e "${frame}"
    printf "%-10s %*s" $(echo -e ${frame}) "$(($width - 1))" '' | tr ' ' _ | indent02
    echo -e "${bodytext}\n"
    #
    # <<--- end function linux_distro -->>
}


function python_version_depcheck(){
    ##
    ## dependency check for a specific version of python binary ##
    ##
    local version
    local version_min="$1"
    local version_max="$2"
    local msg

    local_bin=$(which python3)
    # determine binary version
    version=$($local_bin 2>&1 --version | awk '{print $2}' | cut -c 1-3)

    if (( $(echo "$version > $version_max" | bc -l) )) || (( $(echo "$version <
↪$version_min" | bc -l) )); then

        msg="python version $version detected - must be > $version_min, but <
↪$version_max"
        std_error_exit "$msg" $E_DEPENDENCY

    fi
    #
    # <<-- end function python_depcheck -->>
}


function progress_dots(){
    ##
    ##  Usage:
    ##
    ##      $ long-running-command  &
    ##      $ progress_dots --text "Process XYZ Starting" --end " End xyz"
```

(continues on next page)

(continued from previous page)

```
    ##
    ##      Exists when long-running-command completes
    ##
    ##  Quiet Mode:
    ##      if QUIET = true, runs timer, no stdout printing
    ##
    ##  Dependencies:
    ##      - requires colors.sh (source of indent function)
    ##
    local text
    local endmsg
    local fast
    local width=$(tput cols)
    local stop=$(( $width / 4 ))
    local pid=$!
    local delay="0.1"
    local counter="0"
    local len

    while [ $# -gt 0 ]; do
        case $1 in
            -e | --end)
                endmsg="$2"; shift 2
                ;;

            -f | --fast)
                fast="$2"; shift 2
                ;;

            -t | --text)
                text=$2; shift 2
                ;;
        esac
    done

    if [ ! "$text" ]; then text="Please wait"; fi
    if [ ! "$endmsg" ]; then endmsg="done."; fi

    # print fast dots if short process
    if [ "$fast" = "true" ]; then delay="$(( 1/15 ))"; fi

    # min width of dot pattern
    if [ $stop -lt "80" ]; then stop="80"; fi

    len=${#text}                            # length of text msg, chars
    stopmarker=$stop                        # stop column when not title row
    titlestop=$(( $stop - $len ))           # stop column on text msg row

    # title
    if [[ ! $QUIET ]]; then
        printf -- '\n\n%s' "$text" | indent04
    fi

    # output progress dots
    while [ "$(ps a | awk '{print $1}' | grep $pid)" ]; do

        if [ "$counter" = "0" ]; then
```

(continues on next page)

(continued from previous page)

```
            printf -- '%s' "."
            stop=$titlestop

        elif [ $counter -ge $stop ]; then
            printf -- '\n%s' "." | indent04
            counter="0"
            stop=$stopmarker

        elif [[ ! $QUIET ]]; then
            printf -- '%s' "."
        fi

        sleep $delay
        counter=$(( $counter + 1 ))

    done

    printf -- "  ${endmsg}\n\n"
    #
    # <-- end function ec2cli_spinner -->
    #
}


function python_module_depcheck(){
    ##
    ## validate python library dependencies
    ##
    local check_list=( "$@" )
    local msg

    for module in "${check_list[@]}"; do

        exitcode=$(python3 -c "import $module" > /dev/null 2>&1; echo $?)

        if [[ $exitcode == "1" ]]; then
            # module not imported, not found
            msg="$module is a required python library. Aborting (code $E_DEPENDENCY)"
            std_error_exit "$msg" $E_DEPENDENCY
        fi

    done
    #
    # <<-- end function python_module_depcheck -->>
}


function std_logger(){
    ##
    ##  Summary:
    ##
    ##      std_logger is usually invoked from std_message; ie, all messages
    ##      to stdout are also logged in this function to the log file.
    ##
    ##  Args:
    ##      - msg:      body of the log message text
    ##
```

(continues on next page)

(continued from previous page)

```
    ##      - prefix:   INFO, DEBUG, etc. Note: WARN is handled by std_warn
    ##                  function
    ##
    ##      - log_file: The file to which log messages should be written
    ##
    ##      - version:  Populated if version module exists in
    ##                  pkg_lib. __version__ sourced from within the
    ##                  version module
    ##
    local msg="$1"
    local prefix="$2"
    local log_file="$3"
    local rst=$(echo -e ${RESET})
    local version
    local strip_ansi="false"

    # set prefix if not provided
    if [ ! $prefix ]; then prefix="INFO"; fi

    # set version in logger
    if [ $pkg_lib ] && [ -f $pkg_lib/version.py ]; then
        source "$pkg_lib/version.py"
        version=$__version__

    elif [ "$LIB_VERSION" ]; then
        version=$LIB_VERSION

    elif [ ! "$LIB_VERSION" ]; then
        version="1.0.NA"

    fi

    # write out to log
    if [ ! -f $log_file ]; then

        # create log file
        touch $log_file

        if [ ! -f $log_file ]; then
            echo -e "[$prefix]: $pkg ($version): std_logger failure, $log_file path not writeable"
            exit $E_DIR
        fi

    elif [ "$strip_ansi" = "true" ]; then

        echo -e "$(date +'%Y-%m-%d %T') $host - $pkg - $version - [$prefix]: $msg${rst}" | \
        sed -r "s/\x1B\[([0-9]{1,2}(;[0-9]{1,2})?)?[m|G|K]//g" >> "$log_file"

    else

        echo -e "$(date +'%Y-%m-%d %T') $host - $pkg - $version - [$prefix]: $msg${rst}" >> "$log_file"

    fi
    #
```

(continues on next page)

(continued from previous page)

```
    # <<--- end function std_logger -->>
}


function std_message(){
    ##
    ## Caller formats:
    ##
    ##   Logging to File | std_message "xyz message" "INFO" "/pathto/log_file"
    ##
    ##   No Logging  | std_message "xyz message" "INFO"
    ##
    local msg="$1"
    local prefix="$2"
    local log_file="$3"
    local format="$4"
    local rst=${reset}

    if [ "$4" ]; then format=''; else format='\n'; fi

    if [ "$3" ]; then
        case "$prefix" in
            'ok' | 'OK' | 'DONE')
                std_logger "$msg" "INFO" "$log_file"
                prefix="OK"
                ;;

            'INSTALLED' | 'AVAILABLE' | 'NOT-FOUND')
                filtered=$(echo $msg | sed 's/[|]//g')
                std_logger "$filtered" "INFO" "$log_file"
                ;;

            *)
                # info log message written to log
                std_logger "$msg" "$prefix" "$log_file"
                ;;
        esac
    fi

    if [[ $QUIET ]]; then return 0; fi

    case "$prefix" in
        'ok' | 'OK')
            echo -e "${format}${yellow}[  $green${BOLD}$prefix${rst}${yellow}  ]${rst}
↪  $msg${format}" | indent04
            ;;

        'INSTALLED')
            echo -e "${format}$green${BOLD}$prefix${rst}  |  $msg${format}" | indent04
            ;;

        'AVAILABLE')
            echo -e "${format}$prefix${rst}  |  $msg${format}" | indent04
            ;;

        'FAIL' | 'ERROR' | 'BAD' | 'N/A')
            echo -e "${format}${yellow}[ ${red}${BOLD}$prefix${rst}${yellow} ]${rst}
↪$msg${format}" | indent04
```

(continues on next page)

(continued from previous page)

```
            ;;

        'NOT-FOUND')
            echo -e "${format}${red}${BOLD}$prefix${rst}  |  $msg${format}" | indent04
            ;;

        'WARN')
            echo -e "${format}${yellow}[ ${yellow}$prefix${rst}${yellow} ]${rst}  $msg${format}" | indent04
            ;;

        *)
            echo -e "${format}${yellow}[ $cyan$prefix$yellow ]${rst}  $msg${format}" | indent04
            ;;
    esac
    return 0
    #
    # <<-- end function std_message -->>
}


function std_error(){
    local msg="$1"
    std_logger "$msg" "ERROR" $LOG_FILE
    echo -e "\n${yellow}[ ${red}ERROR${yellow} ]$reset  $msg\n" | indent04
    #
    # <<-- end function std_error -->>
}


function std_warn(){
    local msg="$1"
    local log_file="$2"
    local pc="$(echo -e ${a_brightyellow2})"        # prefix color
    local rst="$(echo -e ${reset})"                 # reset code

    if [ $log_file ]; then
        std_logger "$msg" "WARN" $log_file
    fi

    if [ "$3" ]; then
        # there is a second line of the msg, to be printed by the caller
        echo -e "\n${pv_wgray}[${rst} ${pc}WARN${pv_wgray} ]$reset  $msg" | indent04
    else
        # msg is only 1 line sent by the caller
        echo -e "\n${pv_wgray}[${rst} ${pc}WARN${pv_wgray} ]$reset  $msg\n" | indent04
    fi
    #
    # <<-- end function std_warn -->>
}


function std_error_exit(){
    ##
    ##  standard function presents error msg, automatically
    ##  exits error code
```

(continues on next page)

(continued from previous page)

```
    ##
    local msg="$1"
    local status="$2"
    std_error "$msg"
    exit $status
    #
    # <<-- end function std_warn -->>
}


function timer(){
    ## measure total execution runtime ##
    ##
    ##    Usage:
    ##
    ##        @ beginning:
    ##        $ START=$(timer)
    ##
    ##        @ end time:
    ##        $ printf 'Total runtime: %s\n' $(timer $START)
    ##
    if [[ $# -eq 0 ]]; then

        echo $(date '+%s')

    else
        local  stime=$1
        etime=$(date '+%s')

        if [[ -z "$stime" ]]; then stime=$etime; fi

        dt=$((etime - stime))
        ds=$((dt % 60))
        dm=$(((dt / 60) % 60))
        dh=$((dt / 3600))
        printf '%d:%02d:%02d' $dh $dm $ds

    fi
    #
    # <<-- end function timer -->>
}

# print information about this package
if [ "$pkg" = "std_functions.sh" ]; then
    pkg_info
fi
```

Back to *std_functions.sh*

Back to *Bash Module Index*

CHAPTER 25

Debian Package Creation

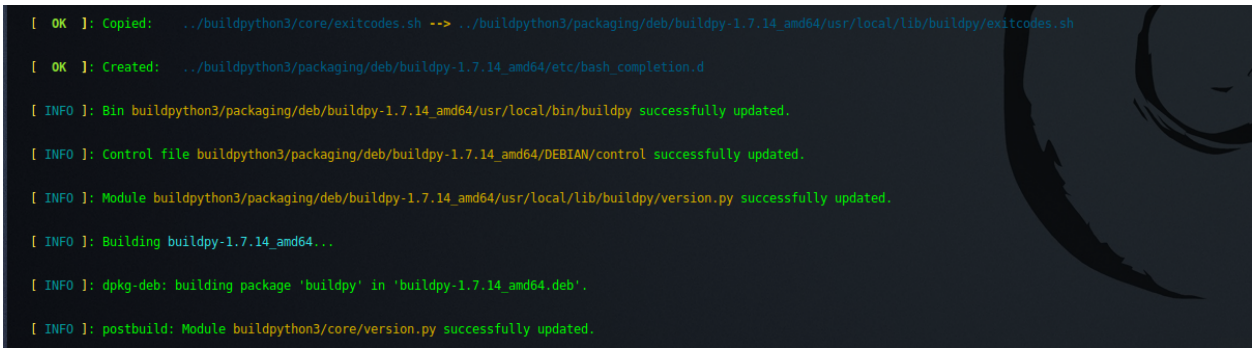
- *Debian Package Creation Start*
- *Debian Package Creation Build*
- *Debian Package Final Contents*

25.1 Debian Package Creation Start

```
$ make builddeb
```

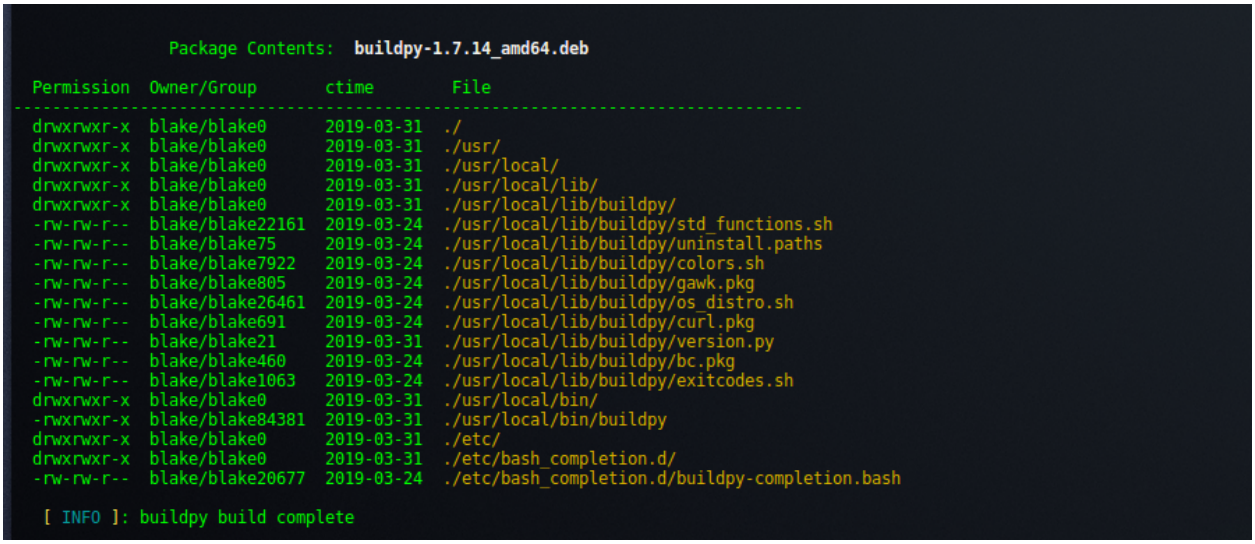
Back to *Debian Package Creation* Index

25.2 Debian Package Creation Build



Back to *Debian Package Creation* Index

25.3 Debian Package Final Contents



Back to *Debian Package Creation* Index

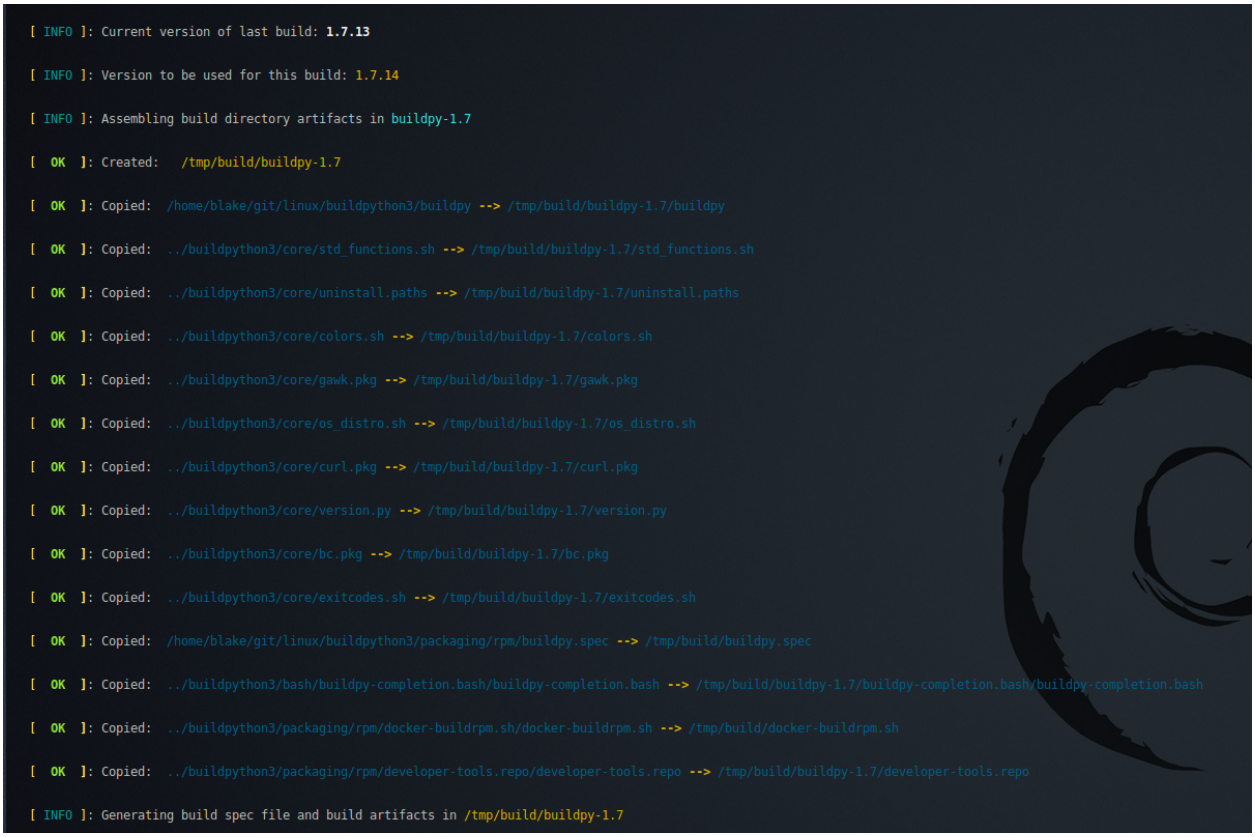
Back to Table Of Contents

CHAPTER 26

RPM Package Creation

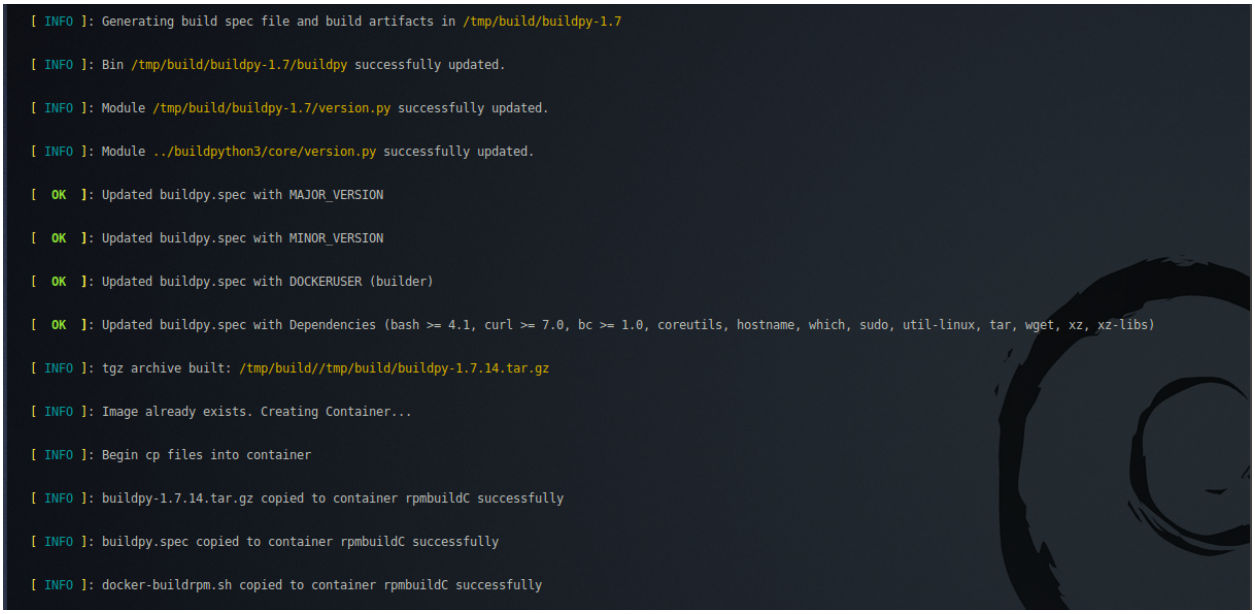
- *RPM Package Assembly*
- *RPM Package Build*
- *Docker RPM Package Build*
- *RPM Package Final Contents*

26.1 RPM Package Assembly



Back to *RPM Package Creation* Index

26.2 RPM Package Build

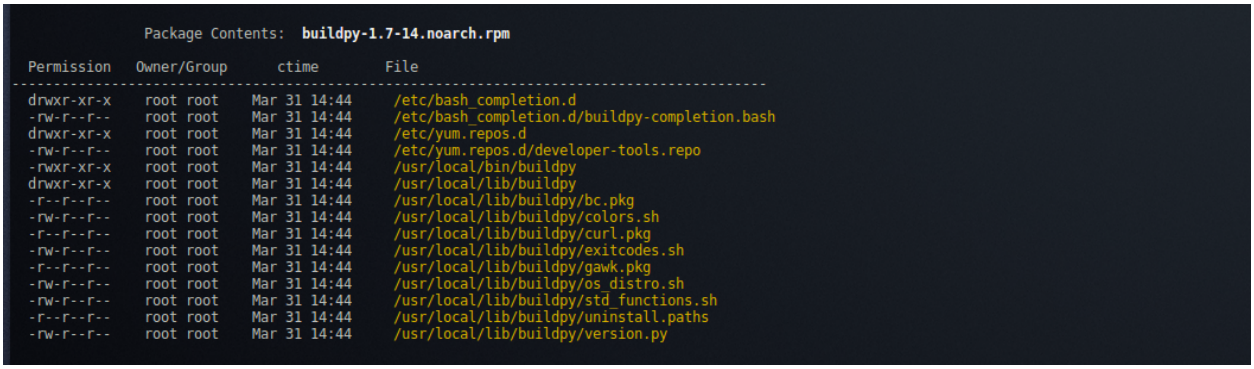


Back to *RPM Package Creation* Index

26.3 Docker RPM Package Build

Back to *RPM Package Creation* Index

26.4 RPM Package Final Contents



```
              Package Contents:  buildpy-1.7-14.noarch.rpm

 Permission   Owner/Group      ctime        File
-------------------------------------------------------------------------------
 drwxr-xr-x    root root    Mar 31 14:44     /etc/bash_completion.d
 -rw-r--r--    root root    Mar 31 14:44     /etc/bash_completion.d/buildpy-completion.bash
 drwxr-xr-x    root root    Mar 31 14:44     /etc/yum.repos.d
 -rw-r--r--    root root    Mar 31 14:44     /etc/yum.repos.d/developer-tools.repo
 -rwxr-xr-x    root root    Mar 31 14:44     /usr/local/bin/buildpy
 drwxr-xr-x    root root    Mar 31 14:44     /usr/local/lib/buildpy
 -r--r--r--    root root    Mar 31 14:44     /usr/local/lib/buildpy/bc.pkg
 -rw-r--r--    root root    Mar 31 14:44     /usr/local/lib/buildpy/colors.sh
 -r--r--r--    root root    Mar 31 14:44     /usr/local/lib/buildpy/curl.pkg
 -rw-r--r--    root root    Mar 31 14:44     /usr/local/lib/buildpy/exitcodes.sh
 -r--r--r--    root root    Mar 31 14:44     /usr/local/lib/buildpy/gawk.pkg
 -rw-r--r--    root root    Mar 31 14:44     /usr/local/lib/buildpy/os_distro.sh
 -rw-r--r--    root root    Mar 31 14:44     /usr/local/lib/buildpy/std_functions.sh
 -r--r--r--    root root    Mar 31 14:44     /usr/local/lib/buildpy/uninstall.paths
 -rw-r--r--    root root    Mar 31 14:44     /usr/local/lib/buildpy/version.py
```

Back to *RPM Package Creation* Index

Back to Table Of Contents

CHAPTER 27

Current Release

27.1 v1.8.12 | Release Notes

Release date: May 21, 2021

Maintenance Release

27.1.1 Maintenance, v1.8.12

Added support for 2-digit Python minor releases with the release of . Previous versions contained a bug which prevented **buildpy** from recognizing Python binary releases higher than the 9 minor version (i.e. higher than 3.9 for example).

(Back to Releases)

CHAPTER 28

Release History

buildpy Release Version	Date
	May 21, 2021
	Dec 27, 2020
	Dec 26, 2020
	Dec 22, 2020
	July 24, 2020
	January 28, 2020
	January 12, 2020
	January 10, 2020
	October 29, 2019
	October 26, 2019
	Sept 26, 2019
	Sept 24, 2019
	August 3, 2019
	April 2, 2019
Release **1.7.14**	April 1, 2019
Release **1.1.13**	March 31, 2019
Release **1.7.8**	December 8, 2019
Release **1.1.0**	January 23, 2019
Release **1.0.1**	December 30, 2017

CHAPTER 29

Release Note Index

29.1 v1.8.11 | Release Notes

Release date: December 27, 2020

Maintenance Release

29.1.1 Maintenance, v1.8.11

1. Added additional Operating System package dependencies to satisfy dependencies during binary compilation.

   Dependencies updated in this release for the following Linux OS variants:

   - (All supported versions).
   - (Version 2).
   - (All supported versions).
2. Added new function to create global `python3` symlink if it does not already exist on the system.

(Back to Releases)

29.2 v1.8.10 | Release Notes

Release date: December 26, 2020

Maintenance Release

29.2.1 Maintenance, v1.8.10

Added additional Operating System package dependencies to satisfy and compression algorithm dependencies required during Python binary compilation.

- **Debian-based Linux OS Packages added:**
  - liblzma-dev | XZ-format compression library - development files
  - liblz-dev | data compressor based on the LZMA algorithm (development)
  - lzma-dev | Compression and decompression in the LZMA format - development files
  - libqdbm-dev | QDBM Database Libraries [development]
  - libgdbm-dev | GNU dbm database routines (development files)
- **Redhat-based Linux OS Packages added:**
  - lzma-sdk | SDK for lzma compression
  - lzma-sdk-devel | Development libraries and headers for lzma-sdk
  - xz-devel | Devel libraries & headers for liblzma
  - gdbm-devel | Development libraries and header files for the GNU Database system

Dependencies updated in this release for the following Linux OS variants:

- (All supported versions).
- (All supported versions).
- (All supported versions).
- (Version 2).
- (All supported versions).

(Back to Releases)

29.3 v1.8.9 | Release Notes

Release date: December 22, 2020

Maintenance Release

29.3.1 Maintenance, v1.8.9

Added additional Operating System package dependencies to satisfy **uuid** requirements during Python binary compilation for the following OS variants:

- (All supported versions).
- (All supported versions).
- (All supported versions).
- (Version 2).
- (All supported versions).

(Back to Releases)

29.4 v1.8.8 | Release Notes

Release date: July 24, 2020

Maintenance Release

29.4.1 Maintenance, v1.8.8

- **20.04 Support**. Added support for newly released (LTS and non-LTS minor releases).
- **20 Support**. Added support for newly released variants such as .

(Back to Releases)

29.5 v1.8.5 | Release Notes

Release date: January 28, 2020

Maintenance Release

29.5.1 Maintenance, v1.8.5

- **Python Version Reconciliation**. Fixed Issue #25/#36: Failure to recognize faulty input for Python version

(Back to Releases)

29.6 v1.8.3 | Release Notes

Release date: January 12, 2020

Feature Release

29.6.1 Feature Updates, v1.8.3

- **Support for CentOS 8 Linux**. Compatibility tested for both 7 and 8 versions of CentOS.
- **Support for Python 3.9**. Python 3.9 now included as an optional major version choice for installation.

(Back to Releases)

29.7 v1.8.1 | Release Notes

Release date: January 10, 2020

Feature Release

29.7.1 Feature Updates, v1.8.1

Install a specific Python binary set. User can select an exact Python binary set to install instead of the latest release. For example, previously, if one wanted to install Python 3.6, and the latest minor revision available was *3.6.8*, only *3.6.8* would be compiled and installed. User can now provide the full revision number when installing and **buildpy** will compile and install that exact revision set as follows:

```
$ sudo buildpy --install Python-3.6.5
```

Alternatively, the following syntax is equivalent:

```
$ sudo buildpy --install 3.6.5
```

(Back to Releases)

29.8 v1.7.20 | Release Notes

Release date: October 29, 2019

Feature Release

29.8.1 Feature Updates, v1.7.20

- User can now choose whether to install release candidate python build sets if downloaded from python.org.
- Issue #58 Resolved: buildpy now checks system-installed python major version before installing. Change prevents installing a compiled python binary executable which is already installed on the system.

(Back to Releases)

29.9 v1.7.19 | Release Notes

Release date: October 26, 2019

Maintenance Release

29.9.1 Maintenance Updates, v1.7.19

- As of release 1.7.18, can now compile and install Python release candidate, Beta, and Alpha code releases. Fix remaining issues with code released in version 1.7.18.

(Back to Releases)

29.10 v1.7.18 | Release Notes

Release date: September 26, 2019

Feature Release

29.10.1 Feature Updates, v1.7.18

- As of release 1.7.18, can now compile and install Python release candidate, Beta, and Alpha code releases. This change was made to enable testing with Python 3.8+ release candidates in the future.

(Back to Releases)

29.11 v1.7.17 | Release Notes

Release date: September 24, 2019

Maintenance Release

29.11.1 Maintenance, v1.7.17

- Release to fix various compatibility issues with modern releases (> version 25)

(Back to Releases)

29.12 v1.7.16 | Release Notes

Release date: August 3, 2019

Functionality & Maintenance Release

29.12.1 Feature Updates, v1.7.16

- Complete rewrite to function *package_info* to enable more info on single screen presentation.
- Bug Fix, Various

(Back to Releases)

29.13 v1.7.15 | Release Notes

Release date: April 2, 2019

Documentation Release

29.13.1 Documentation, v1.7.15

- Multiple updates to Summary and Releases sections.

29.13.2 Maintenance, v1.1.3

- Fixed uninstall operation deleting out-of-band references to python version for removal
- Bug Fix, Various

(Back to Releases)

Table Of Contents

CHAPTER 30

Module Index

- Bash Module Index

CHAPTER 31

Site Index

- genindex

CHAPTER 32

Search Project

- search