

---

# **BcdaMenu Documentation**

***Release 0.g76fcb23.dirty***

**Pete Jemian**

**May 12, 2019**



---

## Contents

---

<b>1</b>	<b>Provides</b>	<b>3</b>
<b>2</b>	<b>Package Information</b>	<b>5</b>
2.1	Usage . . . . .	5
2.2	Contents . . . . .	6
2.3	Indices and tables . . . . .	20
	<b>Python Module Index</b>	<b>21</b>



Creates a GUI menu button to start common beam line software



# CHAPTER 1

---

## Provides

---

- **bcdamenu** : the button menu





---

## Package Information

---

- **author** Pete R. Jemian
- **email** [jemian@anl.gov](mailto:jemian@anl.gov)
- **copyright** 2009-2019, Pete R. Jemian
- **license** ANL OPEN SOURCE LICENSE (see [LICENSE.txt](#) file)
- **documentation** <http://bcdamenu.readthedocs.io>
- **source** <https://github.com/BCDA-APS/bcdamenu>
- **PyPI** <https://pypi.python.org/pypi/bcdamenu>
- **version** 2019.05.0
- **release** 0.g76fcb23.dirty
- **published** May 12, 2019

## 2.1 Usage

### typical

```
user@linux > bcdamenu path/to/settings_file.ini &
```

### bash starter file

```
#!/bin/bash
bcdamenu path/to/settings_file.ini &
```

### usage

```
user@linux > bcdamenu
usage: BcdaMenu [-h] settingsfile
BcdaMenu: error: too few arguments
```

**help**

```
user@linux > bcdamenu -h
usage: BcdaMenu [-h] settingsfile

BcdaMenu: Creates a GUI menu button to start common beam line software

positional arguments:
  settingsfile  Settings file (.ini)

optional arguments:
  -h, --help    show this help message and exit
```

## 2.2 Contents

### 2.2.1 Settings File

The **BcdaMenu** GUI is configured by the content provided in a settings file which name is specified on the command line when starting the program. For example, this Linux command:

```
bcdamenu path/to/menus_settings.ini &
```

#### Version “2017.3.0” format of the Settings file

The settings file version *2017.3.0* uses the *.ini* format, a structure similar to what you would find in Microsoft Windows INI files. This format was chosen for its minimal approach to language markup. The examples provided should guide you to the syntax. For more details, see the documentation for the Python [ConfigParser](#). The [web](#) has many explanations of this informal format.

#### Settings file elements

The settings file consists of sections which are lines starting with “[” and ending with “]”, such as *[section\_name]*. In **BcdaMenu**, these sections are single words with no embedded white space.

Within a section, one or more lines are given with the syntax of *key = value* (or *key: value*). It is expected by the *.ini* format that any key is unique *within* its section. In **BcdaMenu**, the *key* has two parts. First an integer is given that is used to sort the menu’s items in order<sup>4</sup>. The integer is not required to be strictly increasing from 1. Gaps and negative numbers are also allowed. Keep the integers between -9999 .. 9999 to avoid any potential misunderstandings. You *will not* have that many menu items.

Refer to the [Example settings file](#) section for an example settings file. As the examples show, both *key* and *value* are quite flexible strings. A *key* should not contain either the “:” or “=” separator characters. The comment characters allowed by the *.ini* format should also be avoided within either *key* or *value* content.

**{BcdaMenu} section** This section expects the following keys and values. Other keys and values will be ignored.

**title** The window title (default: *BCDA Menu*)

**version** The version of the settings file format. Presently, the only allowed value is *2017.3.0*, the settings format of the initial release. If this format ever changes, this key will be used to identify how to handle the different syntax of the settings file of the new version.

**menus** Names of the sections below with menu specifications. Its value may have more than one menu name, separated by white space.

**menu sections** As referenced by *menus* or *submenu* keys. Each menu section must have a one-word name with no internal white space (to simplify the parsing of names in the *[BcdaMenu]* section. All menu (and submenu) sections must be unique with the settings file. If the same name is used in more than one section, a *configparser.DuplicateSectionError* exception will be raised.

**other sections** will be ignored

## Menu Sections

Each popup menu is configured by the keys and values of a menu section.

**title** Use this as the title on the popup button or submenu name. If the *title* key is not present, the name of the menu section will be used as the text of the popup button.

**separator** draws a horizontal separator line in the menu

**numbered lines** specification of a menu item

- syntax: *index label = command*

where:

**index** an integer used to sort the list numerically (increasing)

**label** text that appears in the menu

- *title* is reserved word and cannot be used as a menu item label
- *separator* is reserved word and cannot be used as a menu item label
- *submenu* Specifies that a submenu will be inserted at this position in the menu. For *submenu*, the *command* gives the name of the menu section to be used.

**NOTE** Do not use the same submenu more than once. This will raise a *ConfigFileKeyError* exception when reading the settings file.

**command** Operating system command to be executed when menu item is chosen. This is a complete command in the operating system syntax that will be started as a separate task by **BcdaMenu**.

Linux: There is no need to append ‘&’ to run the commands in the background.

## Shortcut keys

Shortcut keys are not supported for any menu items.

## Example Settings File

The settings file in the source code distribution ([download](#), also shown below) is an example demonstrating the various features used by **BcdaMenu**.

## file

The example settings file (highlighted lines show the sections, lines 1, 6, & 19 and the specification of the popup menus, line 4) is shown next.

```

1  [BcdaMenu]
2  title = BcdaMenu: 9-ID-C USAXS
3  version = 2017.3.0
4  menus = USAXS linux
5
6  [USAXS]
7  title = 9-ID-C USAXS
8  1 SAXS Imaging tool = /APSShare/epd/rh6-x86/bin/python /home/beams/USAXS/Apps/USAXS_
  ↪dataworker/Main.py
9  2 sample and detector XY position tool = wxmtxy.csh
10 3 separator =
11 4 USAXS Q calculator = qToolUsaxs.csh
12 5 9-ID-C USAXS controls (MEDM) = start_epics
13 6 Save Instr. status to Elog = saveToElog.csh
14 7 PyMca = /APSShare/bin/pymca
15 8 USAXS sample stage tool = /home/beams/USAXS/Apps/wxmtusaxs/wxmtusaxs
16 9 separator =
17 10 submenu = example_submenu
18
19 [linux]
20 1 Xload = xload
21 2 FireFox = firefox
22
23 [example_submenu]
24 title = this is an example of a submenu
25 1 comment = echo "this is not a comment"
26 # this is a comment

```

## screens

This settings file produces a GUI titled *9-ID-C USAXS menu* with two user menus: *USAXS* and *linux*. The following screen views are from Linux.

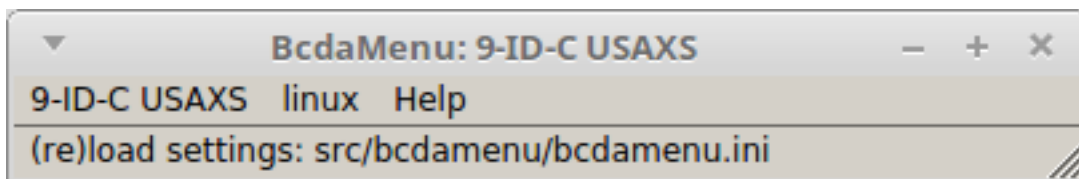


Fig. 1: GUI using example settings file.

This is the *USAXS* menu:

This is the *linux* menu:

### The “Help” popup menu button

The *Help* popup menu button is controlled by the program and is not configurable by the user through the settings file.

This is the *Help* menu:

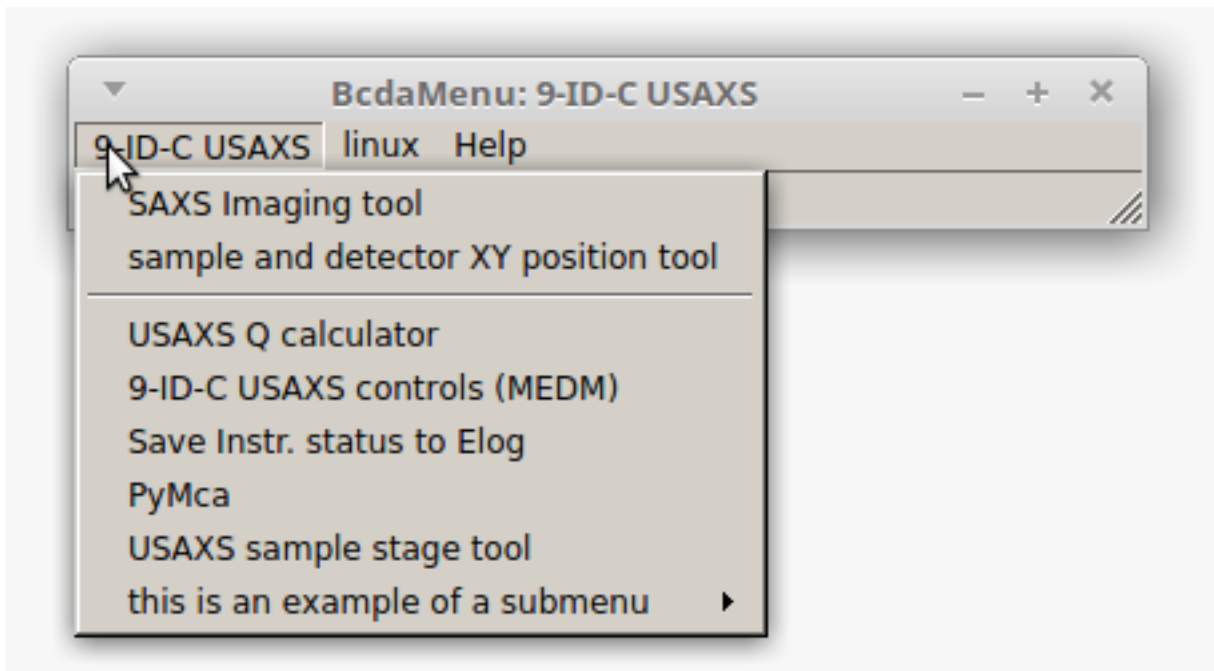


Fig. 2: GUI using example settings file, showing the *USAXS* menu.

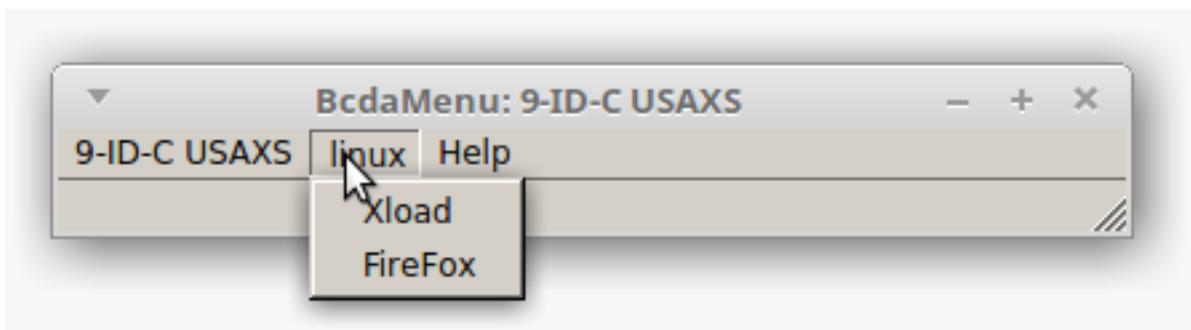


Fig. 3: GUI using example settings file, showing the *linux* menu.

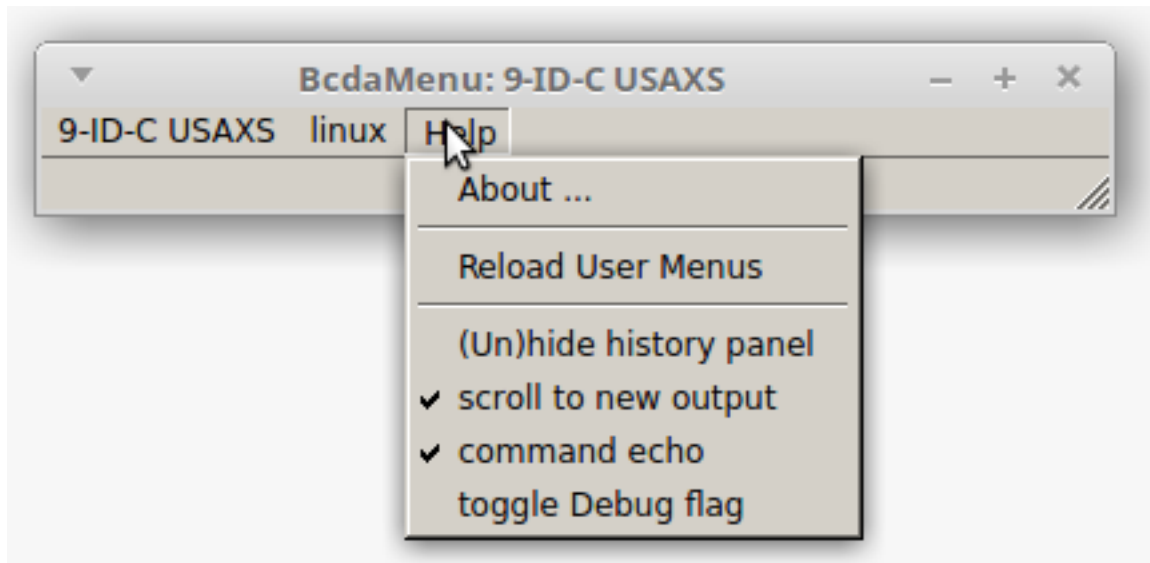


Fig. 4: GUI showing the *Help* menu.

These items are available in the *Help* popup menu:

- *About ...* : prints to the console basic information about this program
- *Reload User Menus* : reloads the settings file (use this when editing/revising that file)
- *(Un)hide history panel* : if checked, show a history panel with command output
- *scroll to new output* : if checked, always scroll when new output is received
- *command echo* : if checked, show command in history when executed
- *toggle Debug flag* : if checked, turns on diagnostic output

## 2.2.2 Examples

### Linux: variety of possibilities

an example from Linux with a number of possibilities

```

1  # bcdamenu.ini
2  #
3  # settings file for BcdaMenu GUI
4
5  [BcdaMenu]
6  title = BcdaMenu: jemian@gov
7  version = 2017.3.0
8  menus = IOC BlueSky linux
9
10 [IOC]
11 title = IOCs on gov
12 20 submenu = iocgov
13 42 submenu = iocprj
14
15 [iocgov]
```

(continues on next page)

(continued from previous page)

```

16 title = IOC: gov (synApps 5.8)
17 ## synApps 5.8 IOC has start/stop/status/console features
18 1 caQtDM iocgov = cd /home/oxygen/JEMIAN/sandbox/ioc/gov; ./start_caQtDM.sh
19 # 2 screen editor = # this is not supported yet
20 10 separator =
21 8 start IOC = cd /home/oxygen/JEMIAN/sandbox/ioc/gov/iocBoot/iocLinux; ./gov.sh_
↳start
22 14 console iocgov = cd /home/oxygen/JEMIAN/sandbox/ioc/gov/iocBoot/iocLinux; gnome-
↳terminal -e "./gov.sh console"
23 15 status iocgov = cd /home/oxygen/JEMIAN/sandbox/ioc/gov/iocBoot/iocLinux; ./gov.sh_
↳status
24 23 stop iocgov = cd /home/oxygen/JEMIAN/sandbox/ioc/gov/iocBoot/iocLinux; ./gov.sh_
↳stop
25
26 [iocprj]
27 title = IOC: prj on gov (synApps 5.6)
28 # synApps 5.6 did not have console & process management
29 101 caQtDM = cd /home/oxygen/JEMIAN/sandbox/ioc/prj; ./start_caQtDM
30 103 start = /home/oxygen/JEMIAN/bin/start_ioc_prj.sh
31 114 console = cd /home/oxygen/JEMIAN/sandbox/ioc/prj/iocBoot/iocLinux; gnome-
↳terminal -e "screen -r"
32
33 [linux]
34 1 edit settings file = /bin/nedit-client /home/oxygen/JEMIAN/bin/bcdamenu.ini
35 2 type settings file = /bin/cat /home/oxygen/JEMIAN/bin/bcdamenu.ini
36 44 xload = xload
37 45 Ku'damm clock = blnuhr
38
39 [BlueSky]
40 title = NSLS-II BlueSky
41 14 BlueSky console = cd /home/oxygen/JEMIAN/Documents; gnome-terminal -e "/home/
↳oxygen/JEMIAN/bin/use_bluesky.sh bluesky"
42 20 submenu = BlueSky-mongodb
43 90 submenu = BlueSky-documentation
44
45 [BlueSky-mongodb]
46 title = mongodb viewer
47 22 start mViewer server (console) = cd /home/oxygen/JEMIAN/Apps/mViewer; gnome-
↳terminal -e "./start_mviewer.sh 8086"
48 24 mongodb viewer in web browser (chrome) = /bin/google-chrome http://localhost:8086/
↳index.html
49 25 mongodb viewer hints = echo "host: gov.aps.anl.gov port: 27017 others leave_
↳blank"
50
51 [BlueSky-documentation]
52 title = web documentation
53 92 BlueSky documentation = /bin/google-chrome http://nsls-ii.github.io/bluesky
54 93 NSLS-II Software documentation = /bin/google-chrome http://nsls-ii.github.io

```

## Windows: caQtDM & PyMca

an example from Windows showing one absolute path and one default path



```

1  [BcdaMenu]
2  title = BcdaMenu: amb (Windows)
3  version = 2017.3.0
4  menus = amb Windows
5
6  [amb]
7  #title =
8  1 CaQtDM = "D:\Program Files\caQTDm\bin\windows-x64\caQtDM.exe"
9  3 separator =
10 7 PyMca = PyMca
11
12 [Windows]
13 10 edit settings file = "D:\Apps\Sublime Text\sublime_text.exe" C:\Users\Pete\.
   ↪ bcdamenu.ini

```

### 2.2.3 History panel in main window

With version 2017.4.0, the *BcdaMenu* was transformed from a *button window* style to a *main window* style of application. This change gained several features:

- keyboard motion between the various menus and menu items
- a status line to report program information (like the *unimenu* predecessor of *BcdaMenu*)
- a main window panel to gather any command output and show history

The history panel (like real history) cannot be changed after it has been written. Both *stderr* and *stdout* from any command are combined and reported in the window. There are options (under the *Help* menu) to control what is written to the history.

Since the original program did not have a history panel, the default is to not display the history panel. Again, the *Help* menu has an item to show/hide the history.

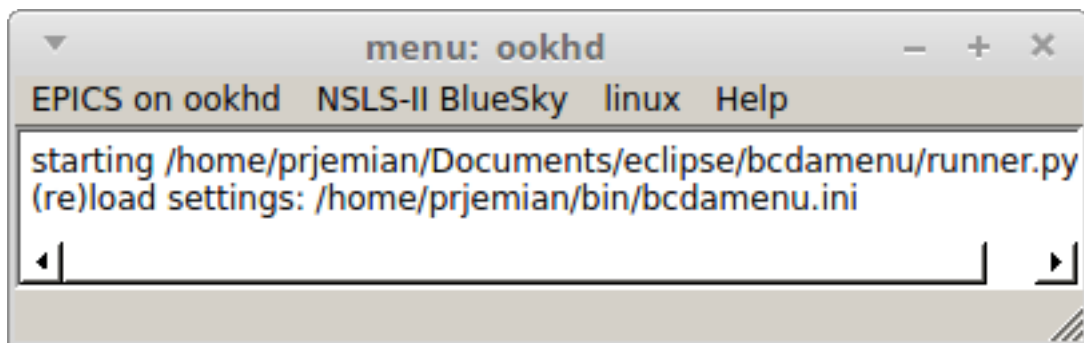


Fig. 5: History panel is shown.



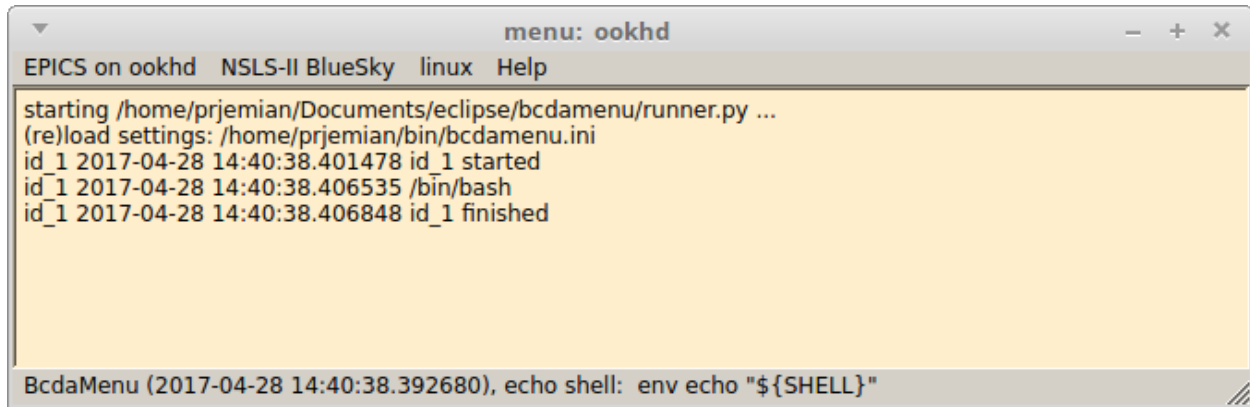


Fig. 6: History panel is shown with debugging turned on. This shows when commands are started and stopped. All lines are time stamped with debugging turned on.

## 2.2.4 Installation

Install this program from the Python Package Index (PyPI) using the *pip* command:

```
pip install bcdamenu
pip install --no-deps bcdamenu
```

The `--no-deps` option tells *pip* not to download and attempt to build newer versions of other required packages.

## Requirements

- Python 2.7
- PyQt4

Your system will need to have the package **PyQt4.QtGui** already installed. A Python 2.7 distribution, such as [Anaconda Python](#) provides this package.

This program was developed on a Windows workstation and tested with various Linux distributions (RHEL7, Linux Mint, Raspberry Pi). It is expected to run on any host that provides the standard Python 2.7 packages *and* **PyQt4**.

## 2.2.5 Source : about

### source code: about

show the About box

Usage (example)

```
ui = about.InfoBox(self)

ui.setTodoURL(__issues__)
ui.setDocumentationURL(__url__)
ui.setLicenseURL(__license_url__)
ui.setTitle(config_file_parser.MAIN_SECTION_LABEL)
ui.setVersionText("software version: " + __version__)
ui.setSummaryText(__doc__.strip())
```

(continues on next page)

(continued from previous page)

```

ui.setAuthorText (__author__)
ui.setCopyrightText (__copyright__)

ui.show()

```

<i>InfoBox</i> ([parent, settings])	a Qt GUI for the About box
<i>myLoadUi</i> (ui_file[, baseinstance])	load a .ui file for use in building a GUI

**class** `bcdamenu.about.InfoBox` (*parent=None, settings=None*)  
 a Qt GUI for the About box

<i>setTodoURL</i> (url)	set the URL for the issue tracker
<i>setDocumentationURL</i> (url)	set the URL for the documentation
<i>setLicenseURL</i> (url)	set the URL for the software license text
<i>setTitle</i> (text)	set the title in the About box
<i>setVersionText</i> (text)	set the version text in the About box
<i>setSummaryText</i> (text)	set the description in the About box
<i>setAuthorText</i> (text)	set the author list in the About box
<i>setCopyrightText</i> (text)	set the copyright string in the About box

**closeEvent** (*event*)  
 called when user clicks the big [X] to quit

**doDocsUrl** ()  
 show documentation URL in default browser

**doIssuesUrl** ()  
 show issues URL in default browser

**doLicense** ()  
 show the license URL in default browser

**doUrl** (*url*)  
 opening URL in default browser

**setAuthorText** (*text*)  
 set the author list in the About box

**setCopyrightText** (*text*)  
 set the copyright string in the About box

**setDocumentationURL** (*url*)  
 set the URL for the documentation

**setLicenseURL** (*url*)  
 set the URL for the software license text

**setSummaryText** (*text*)  
 set the description in the About box

**setTitle** (*text*)  
 set the title in the About box

**setTodoURL** (*url*)  
 set the URL for the issue tracker

**setVersionText** (*text*)

set the version text in the About box

`bcdamenu.about.myLoadUi` (*ui\_file*, *baseinstance=None*, *\*\*kw*)

load a .ui file for use in building a GUI

Wraps `uic.loadUi()` with code that finds our program's *resources* directory.

See <http://nullege.com/codes/search/PyQt4.uic.loadUi>

See <http://bitesofcode.blogspot.ca/2011/10/comparison-of-loading-techniques.html>

inspired by: [http://stackoverflow.com/questions/14892713/how-do-you-load-ui-files-onto-python-classes-with-pyside?](http://stackoverflow.com/questions/14892713/how-do-you-load-ui-files-onto-python-classes-with-pyside?lq=1)  
lq=1

## 2.2.6 Source : `config_file_parser`

source code: `config_file_parser`

parse the configuration file

<code>readConfigFile(file_name)</code>	
<code>ConfigFileError</code>	general exception from <i>config_file_parser</i>
<code>ConfigFileKeyError</code>	exception with a key in the configuration file
<code>clearKnownMenuNames()</code>	keep a list of all known menus so a recursive configuration will be found
<code>MenuBase([parent, order])</code>	base class for menu definitions
<code>Menu([parent, sectionName])</code>	specifications of a menu or submenu
<code>MenuItem([parent, label])</code>	specification of one item in a menu (or submenu)
<code>MenuSeparator([parent])</code>	specification of a separator line in a menu

**exception** `bcdamenu.config_file_parser.ConfigFileError`

general exception from *config\_file\_parser*

**exception** `bcdamenu.config_file_parser.ConfigFileKeyError`

exception with a key in the configuration file

**class** `bcdamenu.config_file_parser.Menu` (*parent=None*, *sectionName=None*)

specifications of a menu or submenu

<code>setTitle(title)</code>	set the text of this menu's title
<code>readConfiguration(config)</code>	read the menu's section from the config file

**readConfiguration** (*config*)

read the menu's section from the config file

**Parameters** **config** (*obj*) – instance of `ConfigParser()`

**setTitle** (*title*)

set the text of this menu's title

**class** `bcdamenu.config_file_parser.MenuBase` (*parent=None*, *order=None*)

base class for menu definitions

**class** `bcdamenu.config_file_parser.MenuItem` (*parent=None*, *label=None*)

specification of one item in a menu (or submenu)

<code>setCommand(command)</code>	set the text of the command to be executed when this menu item is selected
<code>setLabel(label)</code>	set the text to appear in the menu (called in constructor)

**setCommand** (*command*)

set the text of the command to be executed when this menu item is selected

**setLabel** (*label*)

set the text to appear in the menu (called in constructor)

**class** `bcdamenu.config_file_parser.MenuSeparator` (*parent=None*)

specification of a separator line in a menu

`bcdamenu.config_file_parser.clearKnownMenuNames()`

keep a list of all known menus so a recursive configuration will be found

## 2.2.7 Source : launcher

### source code: launcher

BcdaMenu: Creates a GUI menu button to start common software

<code>MainButtonWindow([parent, settingsfilename])</code>	the widget that holds the menu button
<code>CommandThread()</code>	run the command as a subprocess in its own thread, report any output
<code>read_settings(ini_file)</code>	read the user menu settings from the .ini file
<code>gui([settingsfilename])</code>	display the main widget
<code>timestamp()</code>	ISO8601-compliant date & time string
<code>main()</code>	process any command line options before starting the GUI

**class** `bcdamenu.launcher.CommandThread`

run the command as a subprocess in its own thread, report any output

#### Usage

```
process = CommandThread()
process.setName(process_name)
process.setDebug(self.debug)
process.setSignal(self.process_responded)
process.setCommand(command)
process.start()
```

See <https://docs.python.org/3.3/library/subprocess.html>

#### Methods

<code>run()</code>	print any/all output when command is run
<code>execute()</code>	run the command in a shell, reporting its output as it comes in
<code>setCommand(command)</code>	user's command to be run

Continued on next page

Table 7 – continued from previous page

<code>setDebug(value)</code>	<i>True</i> to output more diagnostics
<code>setSignal(signal)</code>	designate the signal to use when subprocess output has been received

**execute()**

run the command in a shell, reporting its output as it comes in

**run()**

print any/all output when command is run

**setCommand(command)**

user's command to be run

**setDebug(value)**

*True* to output more diagnostics

**setSignal(signal)**

designate the signal to use when subprocess output has been received

**class** `bcdamenu.launcher.MainButtonWindow` (*parent=None, settingsfilename=None*)

the widget that holds the menu button

<code>receiver(label, command)</code>	handle commands from menu button
<code>reload_settings_file()</code>	(re)load the settings file and (re)create the menu(s)
<code>build_user_menus(config)</code>	build the user menus
<code>showStatus(text[, isCommand])</code>	write to the status bar
<code>historyUpdate(text)</code>	record history where user can see it
<code>toggleAutoScroll()</code>	change whether (or not) to keep new output in view
<code>toggleDebug([debug_state])</code>	change whether (or not) to output diagnostic information
<code>toggleEcho()</code>	change whether (or not) to echo command before running it
<code>hide_history_window()</code>	toggle the visibility of the history panel
<code>about_box()</code>	display an About box
<code>closeEvent(self, QCloseEvent)</code>	

**about\_box()**

display an About box

**build\_user\_menus(config)**

build the user menus

**closeEvent(self, QCloseEvent)**

**hide\_history\_window()**

toggle the visibility of the history panel

**historyUpdate(text)**

record history where user can see it

**receiver(label, command)**

handle commands from menu button

**reload\_settings\_file()**

(re)load the settings file and (re)create the menu(s)

**showStatus(text, isCommand=False)**

write to the status bar

**toggleAutoScroll()**  
change whether (or not) to keep new output in view

**toggleDebug** (*debug\_state=None*)  
change whether (or not) to output diagnostic information

**toggleEcho()**  
change whether (or not) to echo command before running it

`bcdamenu.launcher.bin(QTextStream) → QTextStream`

`bcdamenu.launcher.gui(settingsfilename=None)`  
display the main widget

`bcdamenu.launcher.hex(QTextStream) → QTextStream`

`bcdamenu.launcher.main()`  
process any command line options before starting the GUI

`bcdamenu.launcher.oct(QTextStream) → QTextStream`

`bcdamenu.launcher.read_settings(ini_file)`  
read the user menu settings from the .ini file

`bcdamenu.launcher.timestamp()`  
ISO8601-compliant date & time string

## 2.2.8 Change History

### Production

#### 2019.05.0

- [#47](#) fix About box for Qt5
- [#49](#) create conda package (Qt5 only)

#### 2017.9.0

- [#43](#) no output to command history
- [#42](#) only one separator line is added, even if multiple are described

#### 2017.4.0

- [#35](#) spawns commands in separate subprocess threads
- [#31](#) capture all console output to history panel
- [#30](#) changed buttons into menubar, added history panel
- [#13](#) About box added
- [#12](#) **submenu** feature added!

#### 2017.3.5

- [#6](#) source code now documented on ReadTheDocs

#### 2017.3.3

- [#27](#) reload settings file now sets window title
- [#26](#) add option to lay out buttons vertical (horizontal is the default)
- [#25](#) reload settings file now clears previous buttons

- #24 started an Examples section in the documentation

### 2017.3.2

- #21 allow command line help to be shown on MS Windows
- #20 show version on command line help
- #18 change internal handling of user menu specs from dictionary to list
- #17 preserve capitalization of menu items

### 2017.3.0

- initial release: 2017.03.21

## Development

### 2017-03-20

- created GitHub repo from *menuUSAXS.py*: <https://github.com/BCDA-APS/bcdamenu>

## 2.2.9 License

Copyright (c) 2009–2019, UChicago Argonne, LLC

All Rights Reserved

BcdaMenu

BCDA, Advanced Photon Source, Argonne National Laboratory

### OPEN SOURCE LICENSE

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Software changes, modifications, or derivative works, should be noted with comments and the author and organization's name.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the names of UChicago Argonne, LLC or the Department of Energy nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
4. The software and the end-user documentation included with the redistribution, if any, must include the following acknowledgment:

"This product includes software produced by UChicago Argonne, LLC under Contract No. DE-AC02-06CH11357 with the Department of Energy."

(continues on next page)

(continued from previous page)

```
*****  
  
DISCLAIMER  
  
THE SOFTWARE IS SUPPLIED "AS IS" WITHOUT WARRANTY OF ANY KIND.  
  
Neither the United States GOVERNMENT, nor the United States Department  
of Energy, NOR uchicago argonne, LLC, nor any of their employees, makes  
any warranty, express or implied, or assumes any legal liability or  
responsibility for the accuracy, completeness, or usefulness of any  
information, data, apparatus, product, or process disclosed, or  
represents that its use would not infringe privately owned rights.  
  
*****
```

## 2.3 Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)



### **b**

`bcdamenu.about`, [13](#)

`bcdamenu.config_file_parser`, [15](#)

`bcdamenu.launcher`, [16](#)



## A

`about_box()` (*bcdamenu.launcher.MainButtonWindow method*), 17

## B

`bcdamenu.about` (*module*), 13

`bcdamenu.config_file_parser` (*module*), 15

`bcdamenu.launcher` (*module*), 16

`bin()` (*in module bcdamenu.launcher*), 18

`build_user_menus()` (*bcdamenu.launcher.MainButtonWindow method*), 17

## C

`clearKnownMenuNames()` (*in module bcdamenu.config\_file\_parser*), 16

`closeEvent()` (*bcdamenu.about.InfoBox method*), 14

`closeEvent()` (*bcdamenu.launcher.MainButtonWindow method*), 17

`CommandThread` (*class in bcdamenu.launcher*), 16

`ConfigFileError`, 15

`ConfigFileKeyError`, 15

## D

`debug`, 12

`doDocsUrl()` (*bcdamenu.about.InfoBox method*), 14

`doIssuesUrl()` (*bcdamenu.about.InfoBox method*), 14

`doLicense()` (*bcdamenu.about.InfoBox method*), 14

`doUrl()` (*bcdamenu.about.InfoBox method*), 14

## E

`execute()` (*bcdamenu.launcher.CommandThread method*), 17

## G

`gui()` (*in module bcdamenu.launcher*), 18

## H

`hex()` (*in module bcdamenu.launcher*), 18

`hide_history_window()` (*bcdamenu.launcher.MainButtonWindow method*), 17

`history`, 12

`historyUpdate()` (*bcdamenu.launcher.MainButtonWindow method*), 17

## I

`InfoBox` (*class in bcdamenu.about*), 14

`installation`, 13

## M

`main()` (*in module bcdamenu.launcher*), 18

`MainButtonWindow` (*class in bcdamenu.launcher*), 17

`Menu` (*class in bcdamenu.config\_file\_parser*), 15

`MenuBase` (*class in bcdamenu.config\_file\_parser*), 15

`MenuItem` (*class in bcdamenu.config\_file\_parser*), 15

`MenuSeparator` (*class in bcdamenu.config\_file\_parser*), 16

`myLoadUi()` (*in module bcdamenu.about*), 15

## O

`oct()` (*in module bcdamenu.launcher*), 18

## P

`pip`, 13

## R

`read_settings()` (*in module bcdamenu.launcher*), 18

`readConfiguration()` (*bcdamenu.config\_file\_parser.Menu method*), 15

`receiver()` (*bcdamenu.launcher.MainButtonWindow method*), 17

`reload_settings_file()` (*bcdamenu.launcher.MainButtonWindow method*), 17

requirements  
    Anaconda Python, 13  
    PyQt4, 13  
    Python 2.7, 13  
run() (*bcdamenu.launcher.CommandThread method*),  
    17

## S

setAuthorText() (*bcdamenu.about.InfoBox method*), 14  
setCommand() (*bcdamenu.config\_file\_parser.MenuItem method*), 16  
setCommand() (*bcdamenu.launcher.CommandThread method*), 17  
setCopyrightText() (*bcdamenu.about.InfoBox method*), 14  
setDebug() (*bcdamenu.launcher.CommandThread method*), 17  
setDocumentationURL() (*bcdamenu.about.InfoBox method*), 14  
setLabel() (*bcdamenu.config\_file\_parser.MenuItem method*), 16  
setLicenseURL() (*bcdamenu.about.InfoBox method*), 14  
setSignal() (*bcdamenu.launcher.CommandThread method*), 17  
setSummaryText() (*bcdamenu.about.InfoBox method*), 14  
settings file, 6  
    example, 7  
    version 2017.3.0, 6  
setTitle() (*bcdamenu.about.InfoBox method*), 14  
setTitle() (*bcdamenu.config\_file\_parser.MenuItem method*), 15  
setTodoURL() (*bcdamenu.about.InfoBox method*), 14  
setVersionText() (*bcdamenu.about.InfoBox method*), 14  
showStatus() (*bcdamenu.launcher.MainButtonWindow method*), 17

## T

timestamp() (*in module bcdamenu.launcher*), 18  
toggleAutoScroll() (*bcdamenu.launcher.MainButtonWindow method*),  
    17  
toggleDebug() (*bcdamenu.launcher.MainButtonWindow method*),  
    18  
toggleEcho() (*bcdamenu.launcher.MainButtonWindow method*), 18