
Buzio Documentation

Release 1.1.0

Chris Maillefaud

Apr 11, 2018

Contents:

1	What is Buzio?	3
1.1	Generate fancy formats	3
1.2	Humanize Python objects	3
1.3	Ask for Input data	4
1.4	Run terminal commands	5
2	Indices and tables	13
	Python Module Index	15

This document will guide you how to install, configure and use [Buzio](#) in your projects.

CHAPTER 1

What is Buzio?

Buzio is a python library tool for printing formatted text in terminal, similar to `termcolor` or `colored`. But unlike these, Buzio has some new features like:

1.1 Generate fancy formats

“Section” example 1:

```
from buzio import console
console.section("First Section")
```

Terminal output:

```
$ >> First Section
$ -----
```

“Section” example 2:

```
from buzio import console
console.section("Main Section", full_width=True, transform="upper center")
```

Terminal output:

```
$ >                               MAIN SECTION                               <
$ -----
```

1.2 Humanize Python objects

Buzio can automatically humanize any python object for printing in terminal:

```
from datetime import datetime, timedelta
from buzio import console

today = datetime.now()
yesterday = today - timedelta(days=1)
my_dict = {
    "start day": yesterday,
    "end day": today
}

console.box(my_dict, date_format="%a, %b-%d-%Y")
```

The output on terminal will be (in blue color):

```
$ *****
$ *
$ *   start day: Thu, Feb-01-2018  *
$ *   end day: Fri, Feb-02-2018   *
$ *
$ *
$ *****
```

1.3 Ask for Input data

You can use **Buzio** to automatically generate “choose” and “select” questions, based on Python objects:

“Choose” example:

```
from buzio import console

my_choices = [
    "Orange",
    "Apple",
    "Potato"
]

console.choose(my_choices)
```

Terminal output:

```
$ 1. Orange
$ 2. Apple
$ 3. Potato
$
$ Select (1-3): ?
```

“Select” example:

```
from buzio import console

my_options = [
    "Save",
    "Save and Exit",
    "Cancel"
]

console.select(my_options)
```


Terminal output:

```
$ Select: (S)ave and Exit, S(A)ve, (C)ancel?
```

You can also “ask” a question and, optionally, use a method to validate the answer:

```
from buzio import console

def check_answer(answer):
    return int(answer) == 4

console.ask("What is the sum of 2+2", validator=check_answer)
print("Thanks!")
```

Terminal output:

```
$ What is the sum of 2+2? : 3
$ Please answer again: 4
$ Thanks!
```

1.4 Run terminal commands

You can use [Buzio](#) to run terminal commands (using Python `subprocess`) and get the *stdout* result:

```
>>> from buzio import console
>>> ret = console.run("echo HelloWorld!", get_stdout=True, verbose=True)
Cmd: echo HelloWorld!
>>> print(ret)
HelloWorld!
```

1.4.1 Tutorial

This document will explain how to install [Buzio](#) and use it in your projects.

Installing Buzio

Install Buzio using the command:

```
$ pip install buzio
```

Importing the Library

```
from buzio import console, formatStr
```

The `console` is a instance of the `Console` class initialized with default color themes. You can also import the class and instantiate with your own settings (See the `:doc:reference` for more info)

The `formatStr` is also a instance of the `Console` class too, but instead of printing in terminal the message, this instance just return the formatted text.

Example:

```
>>> from buzio import console, formatStr
>>> ret = console.ask("What is the sum of 2+2?")
What is the sum of 2+2? : 4
>>> print(ret)
4
>>> ret = formatStr.ask("What is the sum of 2+2")
>>> print(ret)
['\x1b[33mWhat is the sum of 2+2 \x1b[0m']
```

The Output styles

Use the following styles to print your data:

Method	Text Color	Show Prefix
console.box	Fore.CYAN	No
console.error	Fore.RED	Yes
console.info	Fore.CYAN	Yes
console.section	Fore.LIGHTYELLOW_EX	No
console.success	Fore.GREEN	Yes
console.warning	Fore.YELLOW	Yes

The *Text Color* objects are based in [colorama](#) settings. Please check [colorama](#) documentation for all available constants and the [Buzio Package Reference](#) for create your own styles.

The *Show Prefix* column tells if the text to be printed will be the section name append in in. For example, `console.success("Operation Complete")` will be printed as:

```
$ Success: Operation Complete
```

You can control this behavior with the `use_prefix` parameter. For example: `console.info("Starting download...", use_prefix=False)` will print:

```
$ Starting download...
```

Transforming text

You can use the `transform` to format text:

```
>>> console.warning("Hello World", transform="upper")
Warning: HELLO WORLD
>>> console.warning("Hello World", transform="breakline")
Warning:
Hello World
>>> console.warning("Hello World", transform="upper linebreak")
Warning:
HELLO WORLD
>>> console.warning(["Hello", "World"], transform="breakline")
Warning:
Hello
World
```

Current options for transform are:

- upper: UPPERTEXT

- `small`: lowertext
- `title`: Titletext
- `center`: horizontal centering text in terminal
- `breakline`: Break lines in text
- `bold`: Apply Style.BRIGHT effect in text
- `show_counters`: Add counters to text: 1) Hello 2) World (only for dicts and lists)

The Input styles

Use the following styles to print your input data questions:

Method	Text Color	Use Default	Custom Question	Can Validate
<code>console.ask</code>	<code>Fore.YELLOW</code>	Yes	N/A	Yes
<code>console.choose</code>	<code>Fore.LIGHTYELLOW_EX</code>	Yes	Yes	No
<code>console.confirm</code>	<code>Fore.LIGHTMAGENTA_EX</code>	Yes	No	No
<code>console.select</code>	<code>Fore.LIGHTYELLOW_EX</code>	Yes	Yes	No

The *Text Color* objects are based in [colorama](#) settings. Please check [colorama source code](#) for all available constants and the [Buzio Package Reference](#) for create your own styles.

The *Use Default* means you can pass a default value for answer. This can be a python object too. Example: `console.ask("What is the sum of 2+2?", default=4)`

The *Custom Question* means you can pass a custom question for the command. Example: `console.choose(my_list, question="What's your preferred fruit?")`

The *Can Validate* means you can pass a callable object, so [Buzio](#) can validate the prompt answer before return the value to your code. Example: `console.ask("What is the sum of 2+2?", default=4, validator=check_sum)`

Returning data

Data returned from input styles are:

- `console.ask`: string typed.
- `console.choose`: the *python object* selected from original list
- `console.confirm`: boolean
- `console.select`: the *index* for the select object in original list

Special Commands

Use the `console.clear` method to clear the terminal.

Use the `console.progress` to generate a animated progress bar.

Use the `console.run` method to run terminal commands. The return data will be a boolean (if task was succeded) or the capture stdout from command (use `get_stdout=True`). Please check [Buzio Package Reference](#) page for all options.

```
>>> console.run("lsb_release -a", get_stdout=True)
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 17.10
Release:       17.10
Codename:      artful
```

Use the `console.slugify` to generate a slug version from text:

```
.. code-block:: python
```

```
>>> console.slugify("Hello World")
hello_word
```

Use the `console.unitext` to convert text to ascii:

```
.. code-block:: python
```

```
>>> console.unitext("São Paulo")
Sao Paulo
```

1.4.2 Buzio Package Reference

Submodules

buzio.cli module

Buzio main code.

This is the main code for Buzio Package It contains the Console class.

Return

- `console (obj)` = Console instance
- `formatStr (obj)` = Console instance with `format_only=True`

```
class buzio.cli.Console (format_only=False, default_prefix="", default_transform="",
                        default_theme="", theme_dict={'box': '\x1b[36m', 'choose': '\x1b[93m',
'confirm': '\x1b[95m', 'dark': '\x1b[37mx1b[2m', 'error': '\x1b[31m',
'info': '\x1b[36m', 'section': '\x1b[93m', 'success': '\x1b[32m', 'warning':
'\x1b[33m'})
```

Console class.

Attributes: `DEFAULT_THEMES` (Dict): Default color theme `format_only` (bool): Print or format string only
`prefix` (bool): Append prefix on text? `text` (str): text to be formatted/printed `theme` (str): theme selected for
print/format `theme_dict` (dict): theme dictionary loaded `transform` (str): keywords for transform text

```
DEFAULT_THEMES = {'box': '\x1b[36m', 'choose': '\x1b[93m', 'confirm': '\x1b[95m', 'dark': '\x1b[37mx1b[2m', 'error': '\x1b[31m', 'info': '\x1b[36m', 'section': '\x1b[93m', 'success': '\x1b[32m', 'warning': '\x1b[33m'}
```

ask (*obj*, *theme*='warning', *transform*=None, *humanize*=True, *validator*=None, *default*=None, *required*=False, ***kwargs*)
Summary

Args: obj (TYPE): Description theme (str, optional): Description transform (None, optional): Description humanize (bool, optional): Description validator (None, optional): Description default (None, optional): Description required (bool, optional): Description

Returns: TYPE: Description

Raises: ValueError: Description

box (*obj, theme='box', transform=None, humanize=True, **kwargs*)

Function: box Summary: InsertHere Examples: InsertHere

Attributes: Returns: InsertHere

Args: obj (TYPE): Description theme (str, optional): Description transform (None, optional): Description humanize (bool, optional): Description

Returns: TYPE: Description

choose (*choices, question=None, theme='choose', transform=None, humanize=True, default=None, **kwargs*)

Args: choices (TYPE): Description question (None, optional): Description theme (str, optional): Description transform (None, optional): Description humanize (bool, optional): Description default (None, optional): Description

Returns: TYPE: Description

Raises: ValueError: Description

clear ()

Clear terminal.

confirm (*obj=None, theme='confirm', transform=None, humanize=True, default=None, **kwargs*)

Args: obj (None, optional): Description theme (str, optional): Description transform (None, optional): Description humanize (bool, optional): Description default (None, optional): Description

Returns: TYPE: Description

Raises: ValueError: Description

error (*obj, theme='error', transform=None, use_prefix=True, prefix='Error', humanize=True, **kwargs*)

Args: obj (TYPE): Description theme (str, optional): Description transform (None, optional): Description use_prefix (bool, optional): Description prefix (str, optional): Description humanize (bool, optional): Description

Returns: TYPE: Description

info (*obj, theme='info', transform=None, use_prefix=True, prefix='Info', humanize=True, **kwargs*)

Args: obj (TYPE): Description theme (str, optional): Description transform (None, optional): Description use_prefix (bool, optional): Description prefix (str, optional): Description humanize (bool, optional): Description

Returns: TYPE: Description

load_theme (*theme*)

Function: load_theme Summary: InsertHere Examples: InsertHere

Attributes: Returns: InsertHere

Args: theme (TYPE): Description

Raises: ValueError: Description

progress (*count*, *total*, *prefix*=*'Reading'*, *theme*=*None*, *suffix*=*'Complete'*, *barLength*=50, ***kwargs*)

Args: *count* (TYPE): Description *total* (TYPE): Description *prefix* (str, optional): Description *theme* (None, optional): Description *suffix* (str, optional): Description *barLength* (int, optional): Description

Returns: TYPE: Description

run (*task*, *title*=*None*, *get_stdout*=*False*, *run_stdout*=*False*, *verbose*=*False*, *silent*=*False*, *use_prefix*=*True*, *prefix*=*'Cmd'*)
Run command in subprocess.

Args: *task* (string): command to run *title* (string, optional): title to be printed *get_stdout* (bool, optional): return stdout from command *run_stdout* (bool, optional): run stdout before command *verbose* (bool, optional): show command in terminal *silent* (bool, optional): occult stdout/stderr when running command

Bool or String: Task success or Task stdout

section (*obj*, *theme*=*'section'*, *transform*=*None*, *use_prefix*=*False*, *prefix*=*'Section'*, *full_width*=*False*, *humanize*=*True*, ***kwargs*)

Args: *obj* (TYPE): Description *theme* (str, optional): Description *transform* (None, optional): Description *use_prefix* (bool, optional): Description *prefix* (str, optional): Description *full_width* (bool, optional): Description *humanize* (bool, optional): Description

Returns: TYPE: Description

select (*obj*, *theme*=*'choose'*, *humanize*=*True*, *question*=*None*, *default*=*None*, ***kwargs*)
Summary

Args: *obj* (TYPE): Description *theme* (str, optional): Description *humanize* (bool, optional): Description *question* (None, optional): Description *default* (None, optional): Description

Returns: TYPE: Description

Raises: ValueError: Description

slugify (*obj*, *humanize*=*True*, ***kwargs*)
Summary

Args: *obj* (TYPE): Description *humanize* (bool, optional): Description

Returns: TYPE: Description

success (*obj*, *theme*=*'success'*, *transform*=*None*, *use_prefix*=*True*, *prefix*=*'Success'*, *humanize*=*True*, ***kwargs*)

Args: *obj* (TYPE): Description *theme* (str, optional): Description *transform* (None, optional): Description *use_prefix* (bool, optional): Description *prefix* (str, optional): Description *humanize* (bool, optional): Description

Returns: TYPE: Description

untext (*obj*, *theme*=*None*, *transform*=*None*, *humanize*=*True*, ***kwargs*)
Function: untext Summary: InsertHere Examples: InsertHere

Attributes: Returns: InsertHere

Args: *obj* (TYPE): Description *theme* (None, optional): Description *transform* (None, optional): Description *humanize* (bool, optional): Description

Returns: TYPE: Description

warning (*obj*, *theme*=*'warning'*, *transform*=*None*, *use_prefix*=*True*, *prefix*=*'Warning'*, *humanize*=*True*, ***kwargs*)

Args: obj (TYPE): Description theme (str, optional): Description transform (None, optional): Description use_prefix (bool, optional): Description prefix (str, optional): Description humanize (bool, optional): Description

Returns: TYPE: Description

`buzio.cli.get_terminal_size()`

Function: get_terminal_size.

Try to find terminal size using get_terminal_size on Python 3 and 'tput' commands for Python 2. Limited use on Windows: just returns (80, 25)

Tuple of Int: (col, lines)

Module contents

[summary]

[description]

Variables: init() {[type]} – [description] __version__ {str} – [description] console {[type]} – [description] formatStr {[type]} – [description]

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`

b

buzio, [11](#)

buzio.cli, [8](#)

A

`ask()` (buzio.cli.Console method), 8

B

`box()` (buzio.cli.Console method), 9

`buzio` (module), 11

`buzio.cli` (module), 8

C

`choose()` (buzio.cli.Console method), 9

`clear()` (buzio.cli.Console method), 9

`confirm()` (buzio.cli.Console method), 9

`Console` (class in `buzio.cli`), 8

D

`DEFAULT_THEMES` (buzio.cli.Console attribute), 8

E

`error()` (buzio.cli.Console method), 9

G

`get_terminal_size()` (in module `buzio.cli`), 11

I

`info()` (buzio.cli.Console method), 9

L

`load_theme()` (buzio.cli.Console method), 9

P

`progress()` (buzio.cli.Console method), 9

R

`run()` (buzio.cli.Console method), 10

S

`section()` (buzio.cli.Console method), 10

`select()` (buzio.cli.Console method), 10

`slugify()` (buzio.cli.Console method), 10

`success()` (buzio.cli.Console method), 10

U

`unitext()` (buzio.cli.Console method), 10

W

`warning()` (buzio.cli.Console method), 10