

---

# **curses-menu**

**Paul Barrett**

**Oct 25, 2023**



**CONTENTS:**

|          |                                            |           |
|----------|--------------------------------------------|-----------|
| <b>1</b> | <b>Installation</b>                        | <b>1</b>  |
| <b>2</b> | <b>Usage</b>                               | <b>3</b>  |
| 2.1      | Getting a selection . . . . .              | 4         |
| <b>3</b> | <b>API Reference</b>                       | <b>5</b>  |
| 3.1      | CursesMenu — Standard menu class . . . . . | 5         |
| 3.2      | ItemGroup — A group of MenuItems . . . . . | 9         |
| 3.3      | Items . . . . .                            | 9         |
| 3.3.1    | CommandItem . . . . .                      | 9         |
| 3.3.2    | ExitItem . . . . .                         | 10        |
| 3.3.3    | ExternalItem . . . . .                     | 11        |
| 3.3.4    | FunctionItem . . . . .                     | 11        |
| 3.3.5    | MenuItem . . . . .                         | 12        |
| 3.3.6    | SubmenuItem . . . . .                      | 13        |
| 3.4      | Functions . . . . .                        | 13        |
| <b>4</b> | <b>Indices and tables</b>                  | <b>15</b> |
|          | <b>Index</b>                               | <b>17</b> |



## INSTALLATION

All platforms should now just need to run:

```
pip install curses-menu
```



## USAGE

First things first, import the package:

```
import cursesmenu
```

Or just import what you need:

```
from cursesmenu import CursesMenu  
  
from cursesmenu.items import FunctionItem, SubmenuItem, CommandItem
```

Then create a menu:

```
menu = CursesMenu("This is a menu!", "It has a subtitle too!")
```

Create menu items for each choice you need:

```
command_item = CommandItem("Run a console command", "touch hello.txt")  
  
function_item = FunctionItem("Call a function", input, ["Enter some input"])
```

To add other menus as submenus, use a *SubmenuItem*:

```
submenu = CursesMenu("This is the submenu")  
  
submenu_item = SubmenuItem("Show a submenu", submenu, menu=menu)
```

Add the items to the menu:

```
menu.items.append(command_item)  
  
menu.items.append(function_item)  
  
menu.items.append(submenu_item)
```

Then start the menu:

```
menu.start()
```

After that, the menu will spawn its own thread and go about its business. If you want to wait on the user to finish with the menu before continuing, call:

```
menu.join()
```

To combine these two and simply show a menu and immediately wait for the user to exit the menu, call:

```
menu.show()
```

## 2.1 Getting a selection

If you have a list of strings, and you want to allow the user to select one, you can use a `SelectionMenu`:

```
from cursesmenu CursesMenu

a_list = ["red", "blue", "green"]

selection = CursesMenu.get_selection(a_list)
```

Which is equivalent to:

```
from cursesmenu import SelectionMenu

a_list=["red", "blue", "green"]

menu = CursesMenu.make_selection_menu(a_list,"Select an option")

menu.show()

menu.join()

selection = menu.selected_option
```

## API REFERENCE

### 3.1 CursesMenu — Standard menu class

```
class cursesmenu.CursesMenu(title="", subtitle="", *, show_exit_item=True, zero_pad=False,
                             _debug_screens=False)
```

A menu created with the curses library.

#### Parameters

- **title** (str) – The title of the menu
- **subtitle** (str) – The menu subtitle
- **show\_exit\_item** (bool) – Whether the exit item is shown
- **zero\_pad** (bool) – Zero pad the item indices to match the width of the biggest one

#### Variables

- **screen** – The curses window associated with the menu. Created using `curses.newpad` when the menu is started
- **highlight** – Index of the curses color pair used to represent the highlighted item
- **normal** – Index of the curses color pair used to represent other text
- **items** – The list of items for the menu
- **current\_option** – The index of the currently highlighted menu item
- **selected\_option** – The index of the last item the user selected, initially -1
- **should\_exit** – Flag to signal that the menu should exit on its next pass through its main loop
- **returned\_value** – The value returned by the last selected item
- **parent** – The parent menu of this one, or None if this menu is the root menu
- **user\_input\_handlers** – A dictionary mapping character values to functions that handle those characters
- **current\_item** – The MenuItem that's currently highlighted
- **selected\_item** – The Menu item that's currently selected
- **stdscr** – The root curses window
- **menu\_height** – The total height of the menu including the exit item
- **last\_item\_index** – The index of the max item in the menu, including the exit item

- **currently\_active\_menu** – Class variable that holds the currently active menu or None if no menu is currently active (E.G. when switching between menus)

**append\_item(*item*)**

Append an item to the list of items.

**Return type**

None

**start()**

Start the menu's thread and return without blocking.

The menu's thread is a daemon, so if the calling script may exit before the user is finished interacting, use [\*join\(\)\*](#) to block until the menu exits.

**Return type**

None

**join(*timeout=None*)**

Block until the menu exits.

**Parameters**

**timeout** (Optional[int]) – time in seconds until the menu is forced to close

**Return type**

Any

**Returns**

The value returned from the last selected item

**show()**

Start the menu and blocks until it finishes.

**Return type**

Any

**Returns**

The return value from the last selected item

**is\_running()**

Check if the menu has is running (has not been paused).

**Return type**

bool

**Returns**

True if the menu has not been paused false otherwise.

**wait\_for\_start(*timeout=None*)**

Block until the menu starts.

**Parameters**

**timeout** (Optional[int]) – Timeout in seconds

**Return type**

bool

**Returns**

True unless the operation times out

**pause()**

Pause this menu's thread.

**Return type**

None

**resume()**

Resume this menu's thread.

**Return type**

None

**is\_alive()**

Check if the menu's thread is running.

**Return type**

bool

**Returns**

True if the menu's thread is alive, false if not.

**exit(timeout=None)**

Signal the menu to exit and block until it does.

**Parameters**

**timeout** (Optional[int]) – timeout before the menu is forced to close

**Return type**

Any

**Returns**

the value of the last selected item

**draw()**

Draw the menu.

Adds border, title and subtitle, and items, then refreshes the screen.

**Return type**

None

**draw\_item(index, item, index\_text=None)**

Draw an individual item.

**Parameters**

- **index** (int) – The numerical index of the item in the list
- **item** (*MenuItem*) – The item to be drawn
- **index\_text** (Optional[str]) – Text to override the index portion of the item

**Return type**

None

**refresh\_screen()**

Refresh what's onscreen to match the cursor's position.

**Return type**

None

**clear\_screen()**

Clear the screen for this menu.

**Return type**

None

**process\_user\_input()**

Get and then handle the user's input.

**Return type**

int

**Returns**

The character the user input.

**get\_input()**

Get the user's input.

**Return type**

int

**Returns**

The character input by the user.

**select(\_=0)**

Select the current item.

Called for the enter/return key.

**Return type**

None

**go\_to(*user\_input*)**

Go to a given numbered item.

Called on numerical input. Currently implementation only works on single digits.

**Return type**

None

**go\_to\_exit(\_=0)**

Go to the exit item.

Called for Q.

**Return type**

None

**go\_down(\_=0)**

Go down one item, wrap if necessary.

Called when the user presses the down arrow.

**Return type**

None

**go\_up(\_=0)**

Go up one item, wrap if necessary.

Called when the user presses the up arrow.

**Return type**

None

**classmethod** `get_selection(selections, title="", subtitle="")`

Present the user with a menu built from a list of strings and get the index of their selection.

**Parameters**

- **selections** (list[str]) – The list of string possibilities
- **title** (str) – The title of the menu
- **subtitle** (str) – The subtitle of the menu

**Return type**

int

**Returns**

The index in the list of strings that the user selected

**classmethod** `make_selection_menu(selections, title="", subtitle="", *, show_exit_item=False)`

Create a menu from a list of strings.

The return value of the menu will be an index into the list of strings.

**Parameters**

- **selections** (list[str]) – A list of strings to be selected from
- **title** (str) – The title of the menu
- **subtitle** (str) – The subtitle of the menu
- **show\_exit\_item** (bool) – If the exit item should be shown. If it is and the user selects it, the return value will be None

**Return type**

*CursesMenu*

**Returns**

A CursesMenu with items for each selection

## 3.2 ItemGroup — A group of MenuItem

**class** `cursesmenu.ItemGroup(menu, items=None)`

A group of items that belong to a CursesMenu.

Holds the items and ensures that the menu updates when a new one is added. Implements MutableSequence, so should act like a list.

## 3.3 Items

### 3.3.1 CommandItem

Bases: *cursesmenu.items.ExternalItem*

**class** `cursesmenu.items.CommandItem(text, command, arguments=None, menu=None, *, should_exit=False, override_index=None, stdout_filepath=None, **kwargs)`

A menu item that runs a shell command using subprocess.run.

**Parameters**

- **text** (str) – The text for the menu item.
- **command** (str) – The shell command to run when the item is selected.
- **arguments** (Optional[list[str]]) – Additional arguments passed to the command.
- **menu** (Optional[CursesMenu]) – The menu that this item belongs to
- **should\_exit** (bool) – Whether the menu will exit when this item is selected
- **stdout\_filepath** (Optional[PathLike[Any]]) – A filepath that the stdout for the command will be written to
- **kwargs** (Any) – A list of kwargs to be passed to subprocess.run

**action()**

Run the command using subprocess.run.

**Return type**

None

**get\_return()**

Get the exit status of the command or None if it hasn't been run.

**Return type**

Optional[int]

### 3.3.2 ExitItem

Bases: `cursesmenu.items.MenuItem`

**class** `cursesmenu.items.ExitItem`(*menu=None, \*, override\_index=None*)

The exit item for a menu.

Changes representation based on whether the menu is a submenu or the root menu.

**Parameters**

**menu** (Optional[CursesMenu]) – the menu for this item

**show**(*index\_text*)

Get the representation of this item dependent on whether it's in a submenu or the root menu.

**Parameters**

**index\_text** (str) –

**Return type**

str

**Returns**

The representation of this item

### 3.3.3 ExternalItem

Bases: `cursesmenu.items.MenuItem`

**class** `cursesmenu.items.ExternalItem`(*text*, *menu=None*, \*, *should\_exit=False*, *override\_index=None*)

A base class for menu items that need to exit the menu environment temporarily.

**clean\_up()**

Put the console back in curses mode and resume the menu.

**Return type**

None

**set\_up()**

Return the console to its original state and pause the menu.

**Return type**

None

### 3.3.4 FunctionItem

Bases: `cursesmenu.items.ExternalItem`

**class** `cursesmenu.items.FunctionItem`(*text*, *function*, *args=None*, *kwargs=None*, *menu=None*, \*, *should\_exit=False*, *override\_index=None*)

A menu item that executes a Python function with arguments.

**Parameters**

- **text** (str) – The text of the item
- **function** (Callable[... Any]) – A function or lambda to be executed when the item is selected
- **args** (Optional[list[Any]]) – A list of poitional arguments to be passed to the function
- **kwargs** (Optional[dict[Any, Any]]) – A dict of kwargs to be passed to the function
- **menu** (Optional[CursesMenu]) – The menu that this item belongs to
- **should\_exit** (bool) – Whether the menu will exit when this item is selected

**action()**

Call the function with the provided arguments.

**Return type**

None

**get\_return()**

Get the returned value from the function.

**Return type**

Any

**Returns**

The value returned from the function, or None if it hasn't been called.

### 3.3.5 MenuItem

**class** `cursesmenu.items.MenuItem`(*text*, *menu=None*, \*, *should\_exit=False*, *override\_index=None*)

The base class for menu items.

Is displayed in a basic manner and does nothing when selected.

**Parameters**

- **text** (str) – The text representing this menu item
- **should\_exit** (bool) – Whether the menu should exit when this item is selected
- **menu** (Optional[*CursesMenu*]) – The menu that owns this item

**action()**

Do the main action for the item.

If you're just writing a simple subclass, you shouldn't need `set_up` or `clean_up`. The menu just calls them in order. They are provided so you can make subclass hierarchies where the superclass handles some setup and cleanup for its subclasses.

**Return type**

None

**clean\_up()**

Perform cleanup for the item.

**Return type**

None

**get\_return()**

Get the return value for this item.

For a basic MenuItem, just forwards the return value from the menu.

**Return type**

Any

**Returns**

The return value for the item.

**set\_up()**

Perform setup for the item.

**Return type**

None

**show**(*index\_text*)

Provide the representation that should be used for this item in a menu.

The base class is simply “[index] - [text]”

**Parameters**

**index\_text** (str) – The string used for the index, provided by the menu.

**Return type**

str

**Returns**

The text representing the item.

### 3.3.6 SubmenuItem

Bases: `cursesmenu.items.MenuItem`

```
class cursesmenu.items.SubmenuItem(text, submenu=None, menu=None, *, should_exit=False,
                                   override_index=None)
```

A menu item that opens a submenu.

#### Parameters

- **text** (str) – The text of the item
- **submenu** (Optional[[CursesMenu](#)]) – A CursesMenu to be displayed when the item is selected
- **menu** (Optional[[CursesMenu](#)]) – The menu that this item belongs to
- **should\_exit** (bool) – Whether the menu will exit when this item is selected

#### action()

Start the submenu.

#### Return type

None

#### clean\_up()

Block until the submenu is done and then return to the parent.

#### Return type

None

#### get\_return()

Get the returned value from the submenu.

#### Return type

Any

**property menu:** [CursesMenu](#) | None

Get the menu that this item belongs to.

#### set\_up()

Set the screen up for the submenu.

#### Return type

None

**property submenu:** [CursesMenu](#) | None

Get the submenu associated with this item.

## 3.4 Functions

```
cursesmenu.old_curses_menu.parse_old_menu(menu_data)
```

Take an old-style menuData dictionary and return a CursesMenu.

#### Parameters

**menu\_data** (dict) –

#### Returns

A new CursesMenu

**Return type**

*CursesMenu*

## INDICES AND TABLES

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



## A

action() (*cursesmenu.items.CommandItem* method), 10  
 action() (*cursesmenu.items.FunctionItem* method), 11  
 action() (*cursesmenu.items.MenuItem* method), 12  
 action() (*cursesmenu.items.SubmenuItem* method), 13  
 append\_item() (*cursesmenu.CursesMenu* method), 6

## C

clean\_up() (*cursesmenu.items.ExternalItem* method), 11  
 clean\_up() (*cursesmenu.items.MenuItem* method), 12  
 clean\_up() (*cursesmenu.items.SubmenuItem* method), 13  
 clear\_screen() (*cursesmenu.CursesMenu* method), 7  
 CommandItem (class in *cursesmenu.items*), 9  
 CursesMenu (class in *cursesmenu*), 5

## D

draw() (*cursesmenu.CursesMenu* method), 7  
 draw\_item() (*cursesmenu.CursesMenu* method), 7

## E

exit() (*cursesmenu.CursesMenu* method), 7  
 ExitItem (class in *cursesmenu.items*), 10  
 ExternalItem (class in *cursesmenu.items*), 11

## F

FunctionItem (class in *cursesmenu.items*), 11

## G

get\_input() (*cursesmenu.CursesMenu* method), 8  
 get\_return() (*cursesmenu.items.CommandItem* method), 10  
 get\_return() (*cursesmenu.items.FunctionItem* method), 11  
 get\_return() (*cursesmenu.items.MenuItem* method), 12  
 get\_return() (*cursesmenu.items.SubmenuItem* method), 13  
 get\_selection() (*cursesmenu.CursesMenu* class method), 8

go\_down() (*cursesmenu.CursesMenu* method), 8  
 go\_to() (*cursesmenu.CursesMenu* method), 8  
 go\_to\_exit() (*cursesmenu.CursesMenu* method), 8  
 go\_up() (*cursesmenu.CursesMenu* method), 8

## I

is\_alive() (*cursesmenu.CursesMenu* method), 7  
 is\_running() (*cursesmenu.CursesMenu* method), 6  
 ItemGroup (class in *cursesmenu*), 9

## J

join() (*cursesmenu.CursesMenu* method), 6

## M

make\_selection\_menu() (*cursesmenu.CursesMenu* class method), 9  
 menu (*cursesmenu.items.SubmenuItem* property), 13  
 MenuItem (class in *cursesmenu.items*), 12

## P

pause() (*cursesmenu.CursesMenu* method), 6  
 process\_user\_input() (*cursesmenu.CursesMenu* method), 8

## R

refresh\_screen() (*cursesmenu.CursesMenu* method), 7  
 resume() (*cursesmenu.CursesMenu* method), 7

## S

select() (*cursesmenu.CursesMenu* method), 8  
 set\_up() (*cursesmenu.items.ExternalItem* method), 11  
 set\_up() (*cursesmenu.items.MenuItem* method), 12  
 set\_up() (*cursesmenu.items.SubmenuItem* method), 13  
 show() (*cursesmenu.CursesMenu* method), 6  
 show() (*cursesmenu.items.ExitItem* method), 10  
 show() (*cursesmenu.items.MenuItem* method), 12  
 start() (*cursesmenu.CursesMenu* method), 6  
 submenu (*cursesmenu.items.SubmenuItem* property), 13  
 SubmenuItem (class in *cursesmenu.items*), 13

## W

`wait_for_start()` (*cursesmenu.CursesMenu* method),  
[6](#)