**Issue 16: Support ANSI escape codes**

This issue calls for supporting ANSI color coding. However, the ncurses library itself does not support ANSI coloring. Curses does, however, have its own built in colors.

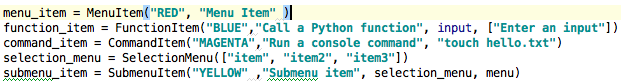
Curses has 8 standard colors:

|  |  |
| --- | --- |
| Color | Curses naming |
| Black | curses.COLOR\_BLACK |
| Red | curses.COLOR\_RED |
| Green | curses.COLOR\_GREEN |
| Yellow | curses.COLOR\_YELLOW |
| Blue | curses.COLOR\_BLUE |
| Magenta | curses.COLOR\_MAGENTA |
| Cyan | curses.COLOR\_CYAN |

These are initialized when you call curses.initscr() and curses.start\_colors(), or you can call curses.wrapper() which will initialize both those functions.

So that a user does not have to dig through to find the function within the curses\_menu.py file where the colors are actually stored, there are 3 global variables at the top of the curses\_menu.py file in which they can just set the background font color, highlighting foreground font color, and the highlighting background font color there. To set the font foreground color for each menu item itself, this must be done in the testing file used to specify your menu build.

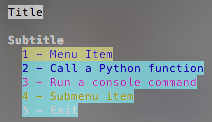
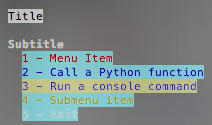
**Take for example:**

*From testing\_curse.py*

*Global variables in curses\_menu.py*



*Your output should look like this:*

*Here you have the text colors varying by item on the menu.*

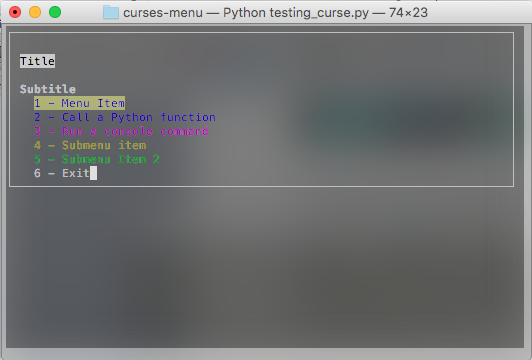
As you can see, those are set in the testing file as the first input for each menu item (excluding the selection menu items) where you build the menu. The background color for each menu item is set globally in the curses\_menu.py file, in most cases a user would generally set it to black or the color of your terminal in which it would appear that only the text color would change. This way it doesn’t interfere with the color scheme of the highlighting colors, but I wasn’t going to assume that a user would never want to change the font background color so user’s choice! The colors set for highlighting are also globally set in the curses\_menu.py file. In this scenario, the highlighting color for text is blue and the background is yellow.

**Issue #25: Filter menu by user input**

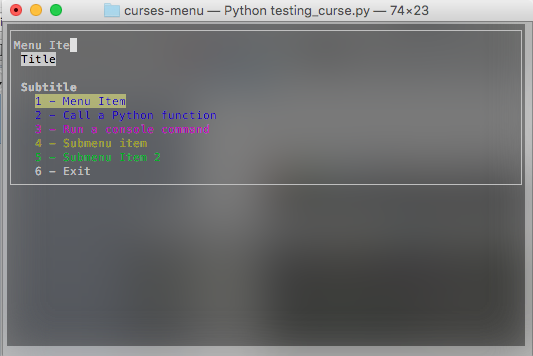
This issue calls for being able to filter the curses menu built from your testing file according to user input. Filtering is automatically enabled when you are in the menu.

**Take for example:**

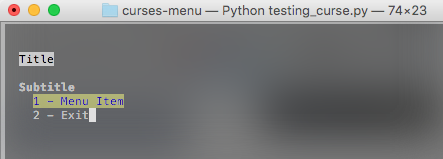
*Here is a menu newly generated, no characters have been input to filter yet.*



*Now we are currently trying to write “Menu Item”, the first item in our menu.*



*Once we complete that string the menu will filter to look like this:*



There are some things to take note of in regards to the filter. The filter does allow backspaces. The cursor will simply move back one and hover over the last character indicating it can be overwritten but the character will not disappear until you enter another character. When backspacing, when you reach the first character in the string the cursor will not hover over it but it can still be overwritten. Once a menu is filtered, there is no way to return to the original menu. If you filter in a submenu you can return to the main curses menu but if you go back into the submenu you will only get the filtered item, not the entire submenu. Once you begin entering characters to filter, they should appear at the top of the menu directly above the title string. However, once you begin to type in a filter string, navigating the menu by numbers is no longer available until the filter string is empty. Also, with regards to the multi-select menu, if you wanted to filter there you must include the [ ] with the name of the item to filter properly.