kirons$_{p}y3_{m}oduleDocumentation$

Release latest

Sep 23, 2018

Contents

1 Installation **3**

1.1 Windows 3

1.2 OS X 3

1.3 Linux 3

2 Window **5**

2.1 Main content 5

2.2 Canvas 6

3 Colors **7**

3.1 The colors (only very basic ones) 7

3.2 RGB and HEX converters 7

4 ConsoleColors **9**

This is a documentation about my python module, kirons_py3_module is an basic library written in python 3 make hard-to-program programs. I am still working on it.

Contents:

CHAPTER 1

Installation

kirons_py3_module is easy to install, just open your terminal/command line or however you call it.

1.1 Windows

Type `pip install kirons-py3-module` to install my library. To update type `pip uninstall kirons-py3-module` and then install it again. No other python libraries needed.

1.2 OS X

Type `pip3 install kirons-py3-module` to install my library. To update type `pip3 uninstall kirons-py3-module` and then install it again. No other python libraries needed.

1.3 Linux

Type `pip3 install kirons-py3-module` to install my library. To update type `pip3 uninstall kirons-py3-module` and then install it again. No other python libraries needed. (same as OS X)

CHAPTER 2

Window

2.1 Main content

This is a little more complex but still very easy. To start I personally like to make a file because this code is going to be used multiple times and I don't like to keep typing. Ok, first you need to import my module and define an window object.

```
from kirons_py3_module import *

screen = Window.init()
```

Now when you run the code it will do nothing, because the window is hidden. You can disable this with `screen.display.show()`

```
from kirons_py3_module import *

screen = Window.init()

screen.display.show()
```

Now that's done we also want to define our own width, height, title and of course we want to disable the ability to resize our screen.

```
from kirons_py3_module import *

display = Window.Display(800,600,"Window example",False,False) # the False values are␣
↪for resizableX and resizableY
screen = Window.init()

screen.set_display(display)
screen.display.show()
```

And finally we want to keep our screen updating. And also I'll explain the code here

```
from kirons_py3_module import * # Most important! load kirons_py3_module.

# Our screen data, params: width, height, title, resizableX, resizableY
display = Window.Display(800,600,"Window example",False,False)

screen = Window.init() # Just creating our screen

screen.set_display(display) # Set our screen data (display) to our screen (screen)
screen.display.show() # Displays our screen

# Keeps our screen updating
while True:
    screen.display.update()
```

2.2 Canvas

By using kirons_py3_module.Window.Display we also defined an Canvas, which currently has `Canvas.canv` and `Canvas.draw_rect`

Canvas.draw_rect(x1,y1,x2,y2,color,outline_color=None) very simple syntax actually, x1 and y1 are the top-left position of the rectangle, x2 and y2 are the bottom-right position. Color is an HEX color use my Colors module (https://kirons-py3-module.readthedocs.io/en/latest/colors.html#the-colors-only-very-basic-ones) to convert RGB to HEX or for some basic colors. outline_color isn't needed, only for an line 1 pixel outside your rectangle.

Example project:

```
from kirons_py3_module import *

display = Window.Display(800,600,"Canvas example",False,False) # the False values are␣
↪for resizableX and resizableY
screen = Window.init()

screen.set_display(display)
screen.display.show()

x = 10
speed = 2

while True:
    screen.canvas.draw_rect(0,0,800,600,Colors.Black)
    screen.canvas.draw_rect(x,10,x+50,60,Colors.Magenta)
    x += speed
    screen.display.update()
```

CHAPTER 3

Colors

kirons_py3_module.Colors is just a simple module I created which contains some colors, a rgb_to_hex converter and a hex_to_rgb

3.1 The colors (only very basic ones)

Colors I added to my kirons_py3_module.Colors module are just really basic ones, an more advanced one is https://ga17-ver.000webhostapp.com/meloonatic_downloads/rpg_tutorials/UltraColor.py which colors mostly aren't supported by my rgb_to_hex and hex_to_rgb. Colors are:

- Black
- White
- Green
- Blue
- Red
- Yellow
- Magenta
- Cyan

I know there aren't much, I told you.

3.2 RGB and HEX converters

rgb_to_hex(rgb) In this case rgb is just a tuple of 3 values between 0 and 255. 1st number = red, 2nd number = green, 3rd number = blue. It can't be explained better. Example code:

```
from kirons_py3_module import *

r = 127
g = 0
b = 255

print(Colors.rgb_to_hex((r,g,b))) # Output: #7f00ff
```

hex_to_rgb(value) Opposite of rgb_to_hex. You give it an HEX color (for ex. #7f00ff), it gives you a tuple of 3 values between 0 and 255. 1st number = red, 2nd number = green, 3rd number = blue. Example code:

```
from kirons_py3_module import *

print(Colors.hex_to_rgb("#7f00ff")) # Output: (127, 0, 255)
```

CHAPTER 4

ConsoleColors

You may confuse this with Colors, Colors are HEX and RGB values of a few colors, but ConsoleColors are ANSI escape codes which contains colors. This module isn't the most advanced one but on http://www.lihaoyi.com/post/BuildyourownCommandLinewithANSIescapecodes.html you can find more colors.

But yeah, my kirons_py3_module.ConsoleColors module contains:

- Reset (to reset the color)
- Black
- Red
- Green
- Yellow
- Blue
- Magenta
- Cyan
- White

Those colors (not Reset) also are available as Bright.

Example:

```
ConsoleColors.Black
ConsoleColors.Bright_blue
ConsoleColors.Reset
```