doscmd-screen Documentation

Release 1.0.3

February 22, 2017

Contents

– Screen positioning and colors in the dos shell (and unix too)

CHAPTER 1

Installation

```
pip install doscmd-screen
```

CHAPTER 2

Changes

Version 1.0.3 introduces thread safe window areas through the Window class.

Changes in version 1.0 include support for non-dos platforms, a visual test script, and zero-based indexing of screen positions. Since the last one is a backwards incompatible change I have upped the major version number. I don't forsee any further backwards incompatible changes in this module.

CHAPTER 3

Documentation

The documentation lives at http://doscmd-screen.readthedocs.org/

CHAPTER 4

Usage

Straight forward positioning and terminal colors in the terminal:

```
import screen  # screen probably needs to be your first import.
scr = Screen()
scr.centerxy(scr.center, scr.middle, '((.))')

scr.writexy(scr.left, scr.bottom,
            'left bottom',
                    color='black', on='red')
```

Works for both Windows..

..and unix-like terminals:

CHAPTER 5

API documentation

screen.Screen

Provides *Screen*, which lets you write text to specific coordinates in the dos command shell, with colors.

class screen.**Screen**(*screeninfo=None*, ***kw*)

Screen provides a interface for positioned writing, with color, to the screen.

bottom

The last (bottom-most) row of the screen.

center

The horizontal center of the screen.

centerxy(*x*, *y*, **args*, ***kw*)

Write text centered around the x coordinate.

cls(*color=None*)

Clear screen, fill it with the given color.

colors = ['black', 'red', 'green', 'yellow', 'blue', 'magenta', 'cyan', 'white']

coords

Mostly a convenience method for debugging.

fill(*x*, *y*, *width*, *height*, *char=' '*, ***kw*)

Fill rectangle with char, and leave the writing position at the beginning of the rectangle (position x,y).

gotoxy(*x*, *y*)

Put cursor at coordinates x, y.

height

The height of the visible portion of the screen buffer.

left

The first column of the screen (the visible part of the screen buffer).

middle

The vertical middle of the screen.

right

The rightmost column of the screen.

rightxy(*x*, *y*, **args*, ***kw*)

Write text right justified at coordinates x, y. The last character will be written at position (x-1, y), which means that e.g.:

```
scr.rightxy(scr.right, scr.bottom, 'bottom right')
```

will be written flush in the bottom right corner, and:

```
scr.rightxy(scr.center, scr.middle, 'hello')
scr.writexy(scr.center, scr.middle, 'world')
```

will output *helloworld* (without a space) in the middle of the screen.

top
The first row of the screen.

width
The width of the visible portion of the screen buffer.

windows (*xcount*, *ycount*)
Returns a list of `count` symetrically created windows.

write (**args*, ***kw*)
Write args at current location, see writexy function for keyword arguments.

writelinesxy (*x*, *y*, **args*, ***kw*)
If the string resulting from prosessing *args* contains newlines, then write the next line at x, y+1, etc.

writexy (*x*, *y*, **args*, ***kw*)
Write args at position x, y. Specify foreground and backround colors with keyword arguments. Available colors:

- black
- red
- green
- yellow
- blue
- magenta
- cyan
- white

(Be aware that the color names can be mapped to entirely different colors by e.g. changing values in the registry: https://github.com/neilpa/cmd-colors-solarized)

class `screen.`**ScreenInfo** (***kw*)
Information about the screen dimensions. Calls SetConsoleMode and GetConsoleScreenBufferInfo on windows, and tries various methods of getting the screen dimension on non-windows platforms.

class `screen.`**Window** (*screen*, *x*, *y*, *width*, *height*)
A window that will scroll text written to it. The screen object is thread safe when used through Window objects.

cls (*color=None*)
Clear window, fill it with the given color.

newline ()

write (**args*)
Write to current position in the window, scrolling the contents as needed.

writexy (*x*, *y*, *txt*)
Write to position x, y relative to the window.

CHAPTER 6

Developing dosbox-screen

Running tests

I'm really not sure how to do any unit testing of this, since the errors are mostly visual.

There is a *test_screen.py* script included that excercises the functionality and creates a screen that can be inspected visually.

Coverage can be run with:

```
coverage run test_screen.py && coverage report
```

Building documentation

```
python setup.py build_sphinx
```

Uploading to PyPI

- only source distribution:

```
    python setup.py sdist upload
```

- source and windows installer:

```
    python setup.py sdist bdist_wininst upload
```

- source, windows, and wheel installer:

```
    python setup.py sdist bdist_wininst bdist_wheel upload
```

- create a documentation bundle to upload to PyPi:

```
    cd build/sphinx/html && zip -r ../../../pypi-docs.zip *
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

S