termtools Documentation

Release 0.0.8

Yasser Mohammad

Feb 06, 2018

CONTENTS

CHAPTER

ONE

A SET OF TERMINAL TOOLS THAT HELPS BUILDING APPS THAT LOOK NICE ON THE TERMINAL

This is a simple library for controlling *text* appearance and placement on terminals. It provides also an advanced progress bar library.

The main advantages of the progress bar library included here are the following:

- Easily show multiple progress bars controlled independently
- Control the placement of the bars anyplace on the screen
- Control the appearance of the bars using a simple markup like interface
- When no information about the actual progress is available (which happens alot in datascience applications), it is possible to just indicate that the system is not stuck

CHAPTER

TWO

TABLE OF CONTENTS

2.1 Reference

This module exposes classes to control displaying text and progress bars at any place of the screen.

The two main classes are TerminalController and ProgressBarController

2.1.1 Functions

humanize_time(secs[, align, ...])	Prints time that is given as seconds in human readable form.
print_progress(iteration, total[, prefix, ...])	Call in a loop to create terminal progress bar

humanize_time

termtools.terminal.**humanize_time**(*secs*, *align=False*, *always_show_all_units=False*)

Prints time that is given as seconds in human readable form. Useful only for times >=1sec.

Parameters

- **float** (*secs*) – number of seconds
- **bool, optional** (*always_show_all_units*) – whether to align outputs so that they all take the same size (not implemented)
- **bool, optional** – Whether to always show days, hours, and minutes even when they are zeros. default False

Returns str formated string with the humanized form

print_progress

termtools.terminal.**print_progress**(*iteration*, *total*, *prefix=''*, *suffix=''*, *decimals=2*, *barLength=50*, *limit_range=True*)

Call in a loop to create terminal progress bar

Parameters

- **int** (*barLength*) – current iteration
- **int** – total iterations
- **str** (*suffix*) – prefix string
- **str** – suffix string
- **int** – positive number of decimals in percent complete
- **int** – character length of bar

2.1.2 Classes

ProgressBarController([barNames, barLength, ...])	A set of progress bars.
TerminalController()	A class for controlling where to print on a screen and the attributes of text to be printed.

ProgressBarController

class termtools.terminal.**ProgressBarController**(*barNames=None*, *barLength=50*, *align_bars=True*)

Bases: object

A set of progress bars.

A set of progress bars with custom pre and post text. It is mostly useful when you have several running threads or suprocesses and each needs its own bar. It allows prefixes and postfixes that can be changed using the following tags: <name> bar name <remaining> remaining time to completion estimate <activity> An indicator that the process represented by the bar is active

Attributes Summary

activityChars
barNames
begTime
completed_background
completed_color
current
last_i
last_n
last_name_updated
over_complete_background
over_complete_color
running_background
running_color
under_complete_background
under_complete_color

Methods Summary

activate([name])	Indicate that the given bar is active.
add_bar(name, *[, i, n, start_timing])	Adds a bar
get_remaining_time(name)	Gets the time remaining till the end of execution (only an estimate)
remove_bar(name)	Removes a bar given its name
set_progress(name, i[, n])	Sets progress for a specific bar, optionally setting its limit as well
show([name, prefix, suffix, bar_length, ...])	Shows a specific bar just in the current place on the screen.
show_all([barname, i, n, clean_screen, ...])	Shows a specific bar or all bars, possibly cleaning the screen
show_in_position([name, prefix, suffix, ...])	Shows the named bar in its appropriate position without touching anything else in the screen

Continued on next page

Table 2.4 – continued from previous page

`start_timing`([forNames])	Starts timing for ETA calculation for the given bar names
`terminal`()	Gets the built-in *TerminalController* object
`update`([name, i, n, prefix, suffix, bar_length])	Updates the progress value of a given bar and shows all the bars in their relative positions

Attributes Documentation

activityChars = ['. ', ' . ', ' . ', ' .', ' . ', ' . ', '. ']

barNames = None

begTime = None

completed_background = 'black'

completed_color = 'green'

current = None

last_i = None

last_n = None

last_name_updated = None

over_complete_background = 'black'

over_complete_color = 'red'

running_background = 'black'

running_color = 'yellow'

under_complete_background = 'black'

under_complete_color = 'red'

Methods Documentation

activate(*name=None*)

Indicate that the given bar is active. If name is None, all bars are indicated to be active by progressing

Parameters **name** (*str*) – bar name. If None is given then all bars are set to be active

Returns self

add_bar(*name*, ***, *i=0*, *n=0*, *start_timing=True*)

Adds a bar

Can control the name, starting progress (i) and total progress

Parameters **name** (*str*) – name of bar

Kwargs: i (int): current progress n (int): total progress start_timing (bool or None): Starts a timer for this bar, otherwise it uses the begTime member which

is common to all bars

Returns self

Remarks:

get_remaining_time(*name*)

Gets the time remaining till the end of execution (only an estimate)

Parameters **name** – bar name

Returns remaining time (ETA)

Return type int

remove_bar(*name*)
Removes a bar given its name

Parameters **name** (*str*) – bar name

Returns self

set_progress(*name*, *i*, *n=None*)
Sets progress for a specific bar, optionally setting its limit as well

Parameters

- **name** (*str*) – bar name
- **i** (*int*) – current progress
- **n** (*int or None*) – Limit (optional)

Returns self

show(*name=None*, *prefix='<name>'*, *suffix='<remaining>'*, *bar_length=-1*, *from_line_beginning=True*)
Shows a specific bar just in the current place on the screen. We should have a new line before it

Args: name (str or None): name of the bar prefix (str): prefix to write before the bar (see the class doc string for possible tag values) suffix (str): postfix (see prefix) bar_length (int): Bar length, if less than zero then the current length set during creation of the bar or

latest setting of its progess will be used.

from_line_beginning (bool): If true, a '

' and flushing will be outputed to set the bar to the beginning

of the line

Returns: self

show_all(*barname=None*, *i=None*, *n=None*, *clean_screen=True*, *prefix='<name><activity>'*, *suffix='<remaining>'*, *bar_length=-1*, *skip_rows=0*)
Shows a specific bar or all bars, possibly cleaning the screen

Parameters

- **barname** (*str or None*) – name of the bar
- **i** (*int or None*) – optional progress value
- **n** (*int or None*) – optional maximum value for the bar
- **clean_screen** (*bool*) – Whether or not to clean the screen before drawing. Default is true
- **prefix** (*str*) – prefix to write before the bar (see the class doc string for possible tag values)
- **suffix** (*str*) – postfix (see prefix)
- **bar_length** (*int*) – Bar length, if less than zero then the current length set during creation of the bar or latest setting of its progess will be used.
- **skip_rows** (*int*) – the number of lines to skip between bars
- **from_line_beginning** (*bool*) – If true, a r and flushing will be outputed to set the bar to the beginning of the line

Returns self

Remarks:

show_in_position(*name=None*, *prefix='<name><activity>'*, *suffix='<remaining>'*, *bar_length=-1*)
Shows the named bar in its appropriate position without touching anything else in the screen

Parameters

- **name** (*str*) – bar name
- **clean_screen** (*bool*) – Whether or not to clean the screen before drawing. Default is true
- **prefix** (*str*) – prefix to write before the bar (see the class doc string for possible tag values)
- **suffix** (*str*) – postfix (see prefix)
- **bar_length** (*int*) – Bar length, if less than zero then the current length set during creation of the bar or latest setting of its progress will be used.

Returns self

start_timing(*forNames=None*)
Starts timing for ETA calculation for the given bar names

Parameters **forNames** (*str or None*) – The list of bar names. If None (Default), the beginning time of all bars is set to now

Returns self

terminal()
Gets the built-in *TerminalController* object

Return type TerminalController

Returns TerminalController

update(*name=None*, *i=None*, *n=None*, *prefix='<name><activity>'*, *suffix='<remaining>'*, *bar_length=-1*)
Updates the progress value of a given bar and shows all the bars in their relative positions

Parameters

- **name** (*str*) – bar name
- **i** (*int or None*) – optional progress value
- **n** (*int or None*) – optional maximum value for the bar
- **clean_screen** (*bool*) – Whether or not to clean the screen before drawing. Default is true
- **prefix** (*str*) – prefix to write before the bar (see the class doc string for possible tag values)
- **suffix** (*str*) – postfix (see prefix)
- **bar_length** (*int*) – Bar length, if less than zero then the current length set during creation of the bar or latest setting of its progess will be used.

Returns self

TerminalController

class termtools.terminal.**TerminalController**
Bases: object

A class for controlling where to print on a screen and the attributes of text to be printed.

Methods Summary

attrib([attrib])	sets the text attributes
background([background])	sets the text background color
bookmark()	saves current cursor position
clear()	clears the screen
color([color])	sets the text foreground color
down([n])	goes down the specified number of rows
eraseDown()	erases all text from currnt line to the end of the screen
eraseLine()	erases all text from currnt line
eraseToBOL()	erases all text from currnt location to the beginning of the line
eraseToEOL()	erases all text from currnt location to the end of the line
eraseUp()	erases all text from currnt line to the beginning of the screen
goto([x, y])	goes to the specified x-y coordingates on the screen
goto_bookmark()	goes to current bookmarked position (must use bookmark() before it)
home([erase_screen])	gets the cursor to the top of the screen
left([n])	goes left the specified number of rows
print(*args, **kwargs)	
printat(txt[, x, y])	goes to the specified x-y coordingates on the screen and prints the text.
reset_attributes()	resets all text attributes to their defaults
right([n])	goes right the specified number of rows
set_attribute([attrib])	sets the text attributes
set_attributes([color, background, attrib])	sets the text attributes to be used by new prints. a value of "keep" keeps the current set
set_background([background])	sets the text background color
set_foreground([color])	sets the text foreground color
up([n])	goes up the specified number of rows

Methods Documentation

attrib(*attrib='keep'*)

sets the text attributes

Parameters **attrib** – one of ['bright','dim','underscore','blink','reverse','hidden']

Return type TerminalController

background(*background='keep'*)

sets the text background color

Parameters **background** – one of ['black','red','green','yellow','blue','magenta','cyan','white']

Return type TerminalController

bookmark()

saves current cursor position

Return type TerminalController

clear()

clears the screen

Return type TerminalController

color(*color='keep'*)

sets the text foreground color

Parameters **color** – one of ['black','red','green','yellow','blue','magenta','cyan','white']

Return type TerminalController

down(*n=1*)

goes down the specified number of rows

Parameters **n** – The number of rows to go down (a negative number goes up)

Return type TerminalController

eraseDown()

erases all text from currnt line to the end of the screen

Return type TerminalController

eraseLine()

erases all text from currnt line

Return type TerminalController

eraseToBOL()

erases all text from currnt location to the beginning of the line

Return type TerminalController

eraseToEOL()

erases all text from currnt location to the end of the line

Return type TerminalController

eraseUp()

erases all text from currnt line to the beginning of the screen

Return type TerminalController

goto(*x=0*, *y=0*)

goes to the specified x-y coordingates on the screen

Parameters

- **x** – x coordinate from the left to right
- **y** – y coordinate from the top to bottom

Return type TerminalController

goto_bookmark()

goes to current bookmarked position (must use bookmark() before it)

Return type TerminalController

home(*erase_screen=False*)

gets the cursor to the top of the screen

Return type TerminalController

left(*n=1*)

goes left the specified number of rows

Parameters **n** – The number of rows to go up (a negative number goes right)

Return type TerminalController

print(**args*, ***kwargs*)

printat(*txt*, *x=0*, *y=0*)

goes to the specified x-y coordingates on the screen and prints the text.

Parameters

- **x** – x coordinate from the left to right
- **y** – y coordinate from the top to bottom

Return type TerminalController

reset_attributes()

resets all text attributes to their defaults

Return type TerminalController

right(*n=1*)

goes right the specified number of rows

Parameters **n** – The number of rows to go up (a negative number goes left)

Return type TerminalController

set_attribute(*attrib='keep'*)

sets the text attributes

Parameters **attrib** – one of ['bright','dim','underscore','blink','reverse','hidden']

Return type TerminalController

set_attributes(*color='keep'*, *background='keep'*, *attrib='keep'*)

sets the text attributes to be used by new prints. a value of "keep" keeps the current set

Parameters

- **color** – one of ['black','red','green','yellow','blue','magenta','cyan','white']
- **background** – one of ['black','red','green','yellow','blue','magenta','cyan','white']
- **attrib** – one of ['bright','dim','underscore','blink','reverse','hidden']

Return type TerminalController

set_background(*background='keep'*)

sets the text background color

Parameters **background** – one of ['black','red','green','yellow','blue','magenta','cyan','white']

Return type TerminalController

set_foreground(*color='keep'*)
sets the text foreground color

Parameters **color** – one of ['black','red','green','yellow','blue','magenta','cyan','white']

Return type TerminalController

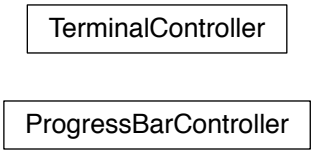
up(*n=1*)

goes up the specified number of rows

Parameters **n** – The number of rows to go up (a negative number goes down)

Return type TerminalController

2.1.3 Class Inheritance Diagram



2.2 Indices and tables

- genindex
- modindex
- search

INDEX

S

T

U