pressenter2exit Documentation

Release 0.0.9

Author

Oct 02, 2017

Contents:

CHAPTER 1

pressenter2exit package

Module contents

class pressenter2exit.**PressEnter2Exit**(*message='Enter pressed'*)

Bases: threading.Thread

Press Enter to Exit class. Facilitates exit of a Python CLI program in a controlled way.

get_duration()

time since instantiation

Returns time since instantiation (in seconds)

get_enter_duration()

time from instantiation to when enter was pressed

Returns time from instantiation to when enter was pressed or None if enter not yet pressed (in seconds)

get_reaction_time()

get how long it took the program that uses this class to react to the pressing of enter

Returns the time from when enter was pressed to now (in seconds)

get_start_time()

get time of instantiation (in seconds since epoch)

Returns time of instantiation (in seconds since epoch)

run()

run method

CHAPTER 2

pressenter2exit (Press Enter To Exit)

- genindex
- modindex
- search

pressenter2exit ("Press Enter To Exit") facilitates long-running CLI programs to exit in clean and controlled way.

Documentation: pressenter2exit.readthedocs.io.

CHAPTER 3

The Problem

Long running CLI (command line interface) programs can be useful, but there is often a need to exit them in a clean manner (as opposed to, for example, pressing ctrl-c).

Examples

- Programs that 'crunch' on a large number of input data sets, but you want to be able to cleanly interrupt the program after it's done with the current data set.
- Programs that run in an 'infinite loop' waiting for some sort of input data to appear. You'd like to be able to exit the 'infinite loop' in a controlled way.
- Long-running system tests where you want to run, say, overnight but when you return you want to exit in a controlled way so you can get proper status and statistics (as opposed to ctrl-c).

The benefit to exiting in a controlled way is that you don't end up with a data set partially processed, in an unknown state, and/or be unable to output statistics from the run. By using pressenter2exit, processing that has been done so far can be useful and not merely thrown away, which can happen if a program is forcefully and immediately aborted.

Example Code

```
from time import time

from pressenter2exit import PressEnter2Exit

exit_control = PressEnter2Exit()

start_time = time()
# loops until enter is pressed or we reach the end of our run time
while exit_control.is_alive() and time() - start_time < 20.0:
    print("I've been waiting for %f seconds." % (time() - start_time))
```

```
    exit_control.join(4.0)  # use join() instead of time.sleep() to ensure an␣
↪immediate exit
print('Done! Exiting after %f seconds.' % (time() - start_time))
```

Output when enter is pressed mid-run:

```
Press enter to exit:
I've been waiting for 0.000005 seconds.
I've been waiting for 4.000108 seconds.
Done! Exiting after 5.331482 seconds.
```

Output when the program is allowed to finish on its own:

```
Press enter to exit:
I've been waiting for 0.000006 seconds.
I've been waiting for 4.004918 seconds.
I've been waiting for 8.007299 seconds.
I've been waiting for 12.009813 seconds.
I've been waiting for 16.012574 seconds.
Done! Exiting after 20.016182 seconds.
```

Platform Support

pressenter2exit doesn't use any platform specific libraries, so it should run on any platform that supports the CLI (e.g. Windows, MacOS, Linux, etc.).

Python Module Index

p

Index

G

P

R