Kartikeya Agarwal
2019UCO1692

Non-Blocking Subroutine without Explicit Timer

```
import time
import threading
import os
import signal


class Timer(threading.Thread):
    _timeout = False
    _timer = 0
    _stopped = False

    def __init__(self, delay):
        super(Timer, self).__init__()
        self.restart(delay)

    def is_timeout(self):
        return self._timeout

    def stop(self):
        self._stopped = True

    def restart(self, delay):
        self._stopped = False
        self._timer = time.time() + delay

    def run(self):

        while not self._stopped:
            time.sleep(0.1)
            if time.time() >= self._timer:
                break
        if not self._stopped:
            self._timeout = True

            # check os name
            if os.name == 'nt':
                # we are on Windows
                os.kill(os.getpid(), signal.CTRL_C_EVENT)
            else:
                # we are on a Posix/Unix (or very unlikely on java) system
                os.kill(os.getpid(), signal.SIGINT)
```

Kartikeya Agarwal
2019UCO1692

```
def main():
    first_input = input('First input:')

    delay = 10
    timer = Timer(delay)
    timer.daemon = True

    try:
        print('\nStarting the timer for the second input %r second(s)' % delay
)
        timer.start()

        second_input = input('Second input:')

        print('\nWell done. Stopping the timer!\n')
        timer.stop()

        print('Input values: %r %r\n' % (first_input, second_input))

        # do your stuff here...

    except KeyboardInterrupt:
        if timer.is_timeout():
            print("\nTimeout!")
        else:
            print("\nUser interrupted the input")


main()
```

OUTPUT

```
First input:13

Starting the timer for the second input 10 second(s)
Second input:2

Well done. Stopping the timer!

Input values: '13' '2'
```