**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)

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**

