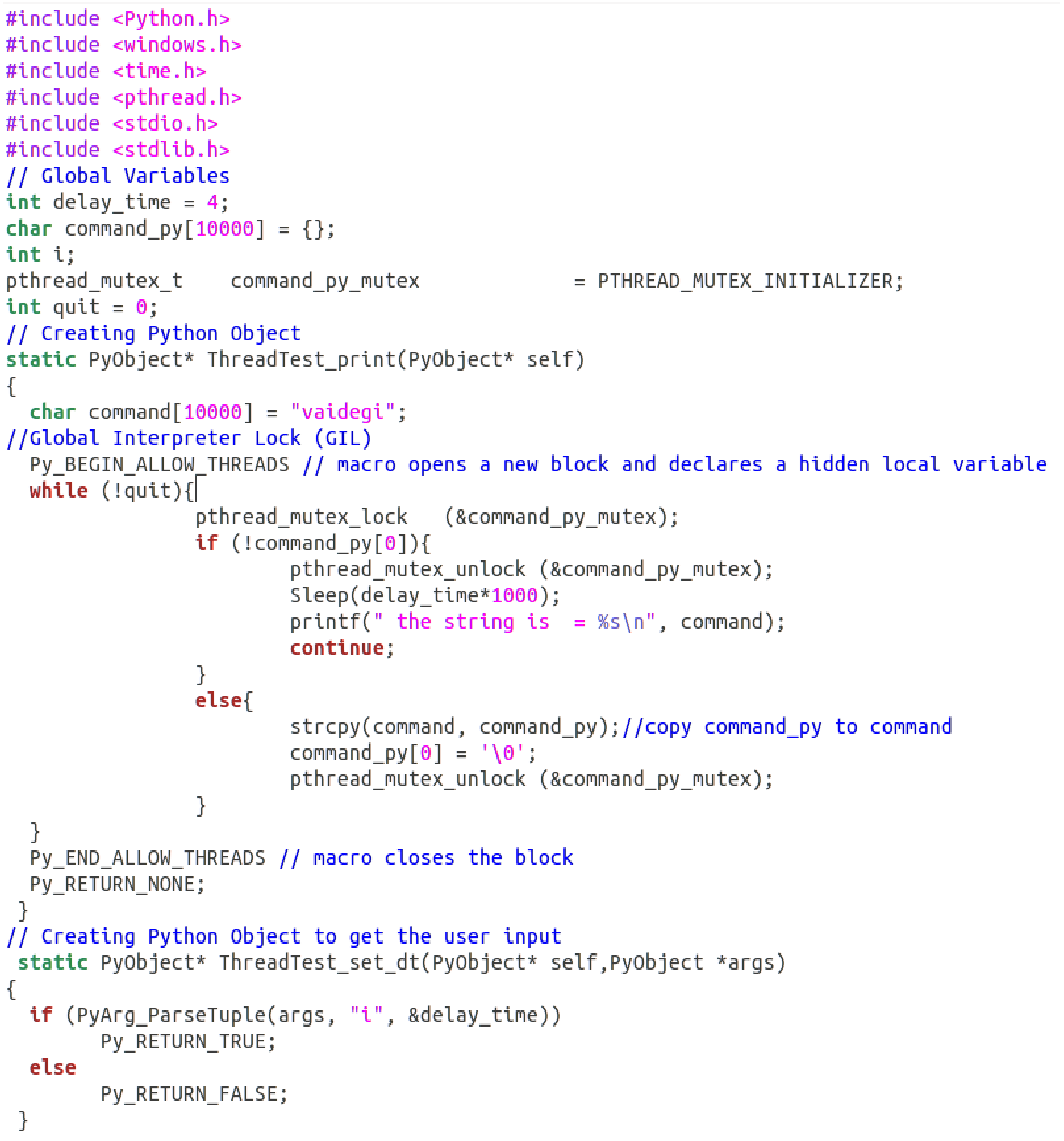
Calling C Function From Python Using Thread

Aim:

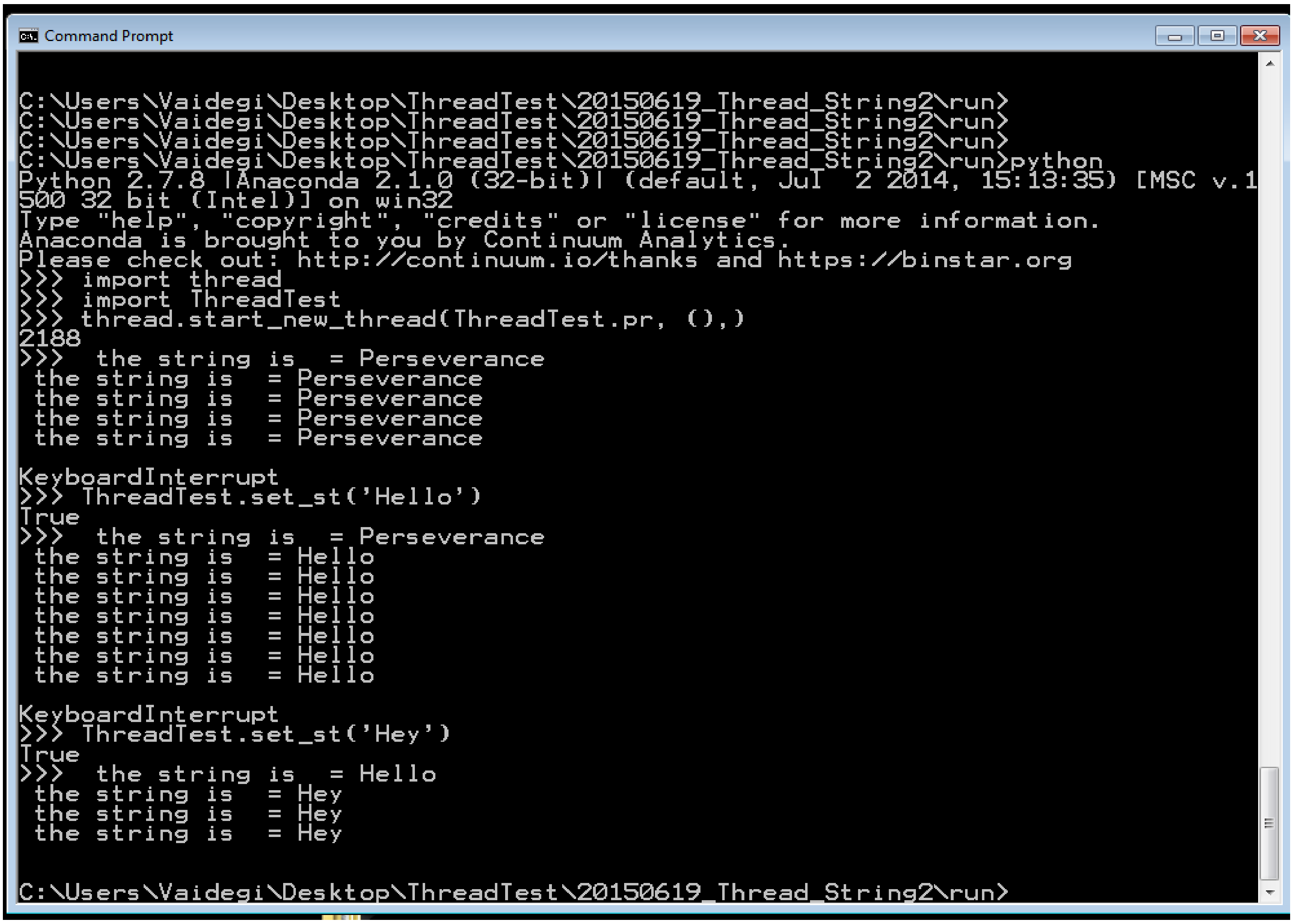
This is an example for parallel processing. So I made a program to print a string after every 4 seconds defaultly. But the goal is to modify the default time of printing(*delay_time*) & string to be printed (*command*) without stoping the process (while the program is running). There arise a concept of thread.



```c
#include <Python.h>
#include <windows.h>
#include <time.h>
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
// Global Variables
int delay_time = 4;
char command_py[10000] = {};
int i;
pthread_mutex_t    command_py_mutex             = PTHREAD_MUTEX_INITIALIZER;
int quit = 0;
// Creating Python Object
static PyObject* ThreadTest_print(PyObject* self)
{
  char command[10000] = "vaidegi";
//Global Interpreter Lock (GIL)
  Py_BEGIN_ALLOW_THREADS // macro opens a new block and declares a hidden local variable
  while (!quit){
                pthread_mutex_lock   (&command_py_mutex);
                if (!command_py[0]){
                        pthread_mutex_unlock (&command_py_mutex);
                        Sleep(delay_time*1000);
                        printf(" the string is  = %s\n", command);
                        continue;
                }
                else{
                        strcpy(command, command_py);//copy command_py to command
                        command_py[0] = '\0';
                        pthread_mutex_unlock (&command_py_mutex);
                }
  }
  Py_END_ALLOW_THREADS // macro closes the block
  Py_RETURN_NONE;
 }
// Creating Python Object to get the user input
 static PyObject* ThreadTest_set_dt(PyObject* self,PyObject *args)
{
  if (PyArg_ParseTuple(args, "i", &delay_time))
        Py_RETURN_TRUE;
  else
        Py_RETURN_FALSE;
 }
```

```c
// Creating Python Object to get the user input
 static PyObject* ThreadTest_set_st(PyObject* self,PyObject *args)
{
    char* command_local; // It is more safe to use character pointer than character array.
        if (PyArg_ParseTuple(args, "s", &command_local)){
                pthread_mutex_lock   (&command_py_mutex);
                strcpy(command_py, command_local);//copy command_local to command_py
                pthread_mutex_unlock (&command_py_mutex);
                Py_RETURN_TRUE;
        }
        else{
                Py_RETURN_FALSE;
        }

 }

  static PyObject* ThreadTest_quit(PyObject* self)
{
   quit = 1;
 }
static char ThreadTest_print_docs[] =
    "ThreadTest_pr( ) :prints hello world every 'delay_time ' seconds\n";

static char ThreadTest_set_docs[] =
    "ThreadTest_set_docs( ) :sets 'delay_time ' seconds\n";

static char ThreadTest_set1_docs[] =
    "ThreadTest_set1_docs( ) :sets 'command ' seconds\n";

static char ThreadTest_quit_docs[] =
    "ThreadTest_quit( ) :       quits program\n";

static PyMethodDef ThreadTest_funcs[] =
{
  {"pr", (PyCFunction)ThreadTest_print, METH_NOARGS, ThreadTest_print_docs},
  {"set_dt", (PyCFunction)ThreadTest_set_dt, METH_VARARGS, ThreadTest_set_docs},
  {"set_st", (PyCFunction)ThreadTest_set_st, METH_VARARGS, ThreadTest_set1_docs},
  {"q", (PyCFunction)ThreadTest_quit, METH_NOARGS, ThreadTest_quit_docs},
  {NULL}
};

void initThreadTest(void)
{
    Py_InitModule3("ThreadTest", ThreadTest_funcs,
                   "Extension module ThreadTest");
}
```

Command Prompt:


Command Prompt

```
C:\Users\Vaidegi\Desktop\ThreadTest\20150619_Thread_String2\run>
C:\Users\Vaidegi\Desktop\ThreadTest\20150619_Thread_String2\run>
C:\Users\Vaidegi\Desktop\ThreadTest\20150619_Thread_String2\run>
C:\Users\Vaidegi\Desktop\ThreadTest\20150619_Thread_String2\run>python
Python 2.7.8 |Anaconda 2.1.0 (32-bit)| (default, Jul  2 2014, 15:13:35) [MSC v.1
500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://binstar.org
>>> import thread
>>> import ThreadTest
>>> thread.start_new_thread(ThreadTest.pr, (),)
2188
>>>  the string is   = Perseverance
 the string is   = Perseverance
 the string is   = Perseverance
 the string is   = Perseverance
 the string is   = Perseverance

KeyboardInterrupt
>>> ThreadTest.set_st('Hello')
True
>>>  the string is   = Perseverance
 the string is   = Hello
 the string is   = Hello
 the string is   = Hello
 the string is   = Hello
 the string is   = Hello
 the string is   = Hello
 the string is   = Hello

KeyboardInterrupt
>>> ThreadTest.set_st('Hey')
True
>>>  the string is   = Hello
 the string is   = Hey
 the string is   = Hey
 the string is   = Hey


C:\Users\Vaidegi\Desktop\ThreadTest\20150619_Thread_String2\run>
```