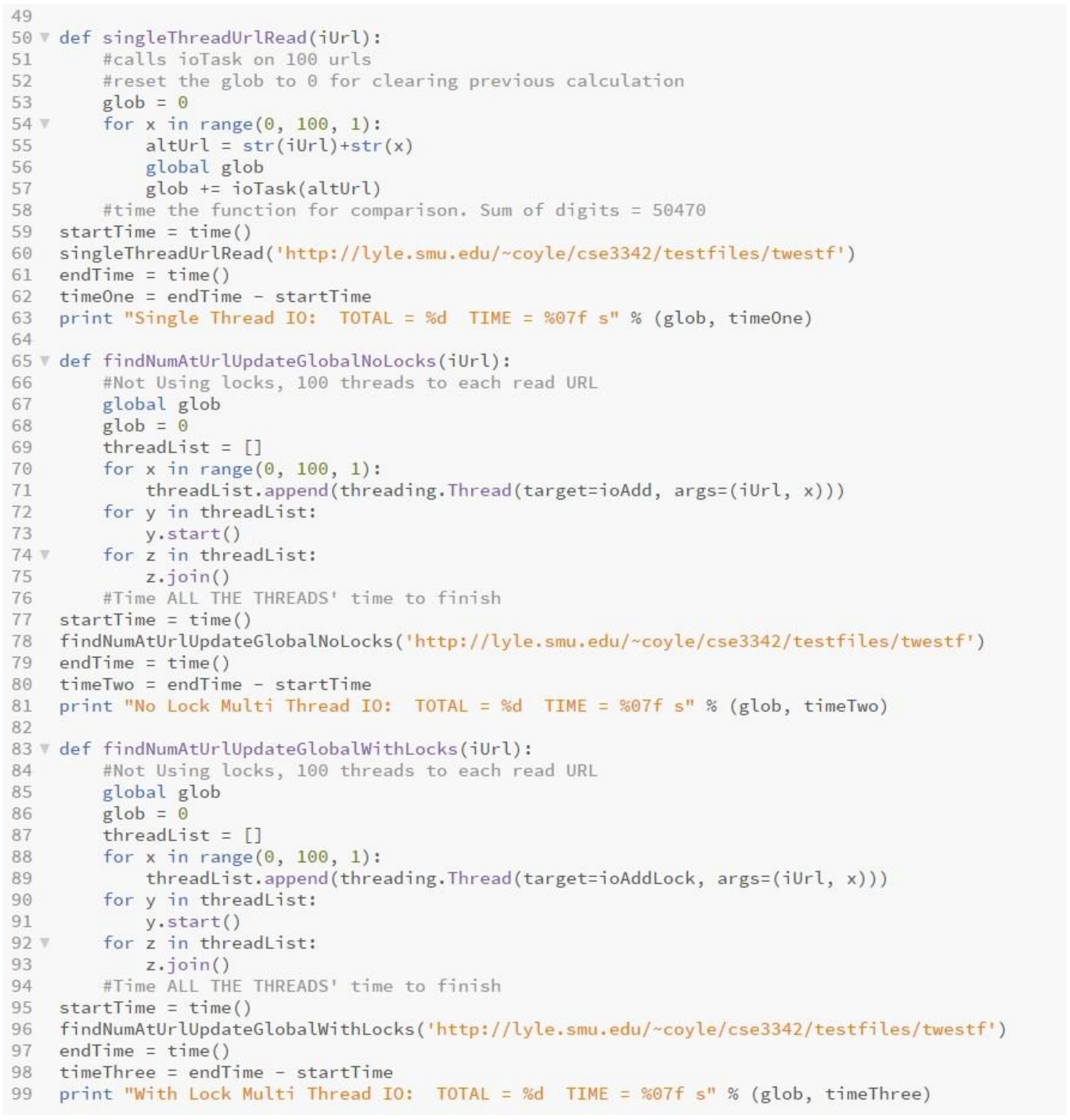
Seung Ki Lee

CSE 3342 HW 10

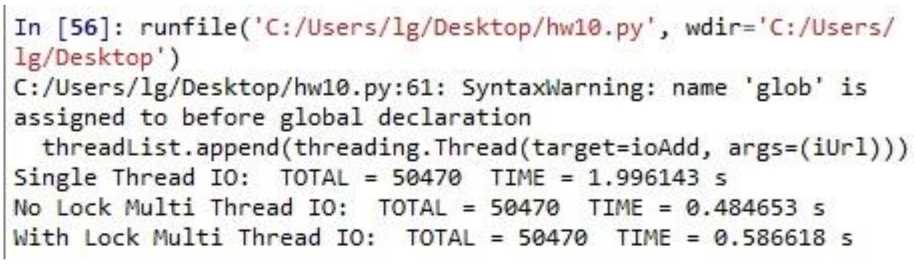
Code

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 10 16:50:57 2017

@author: lg
"""

import matplotlib.pyplot as plot
import threading
import urllib
from timeit import default_timer as time

glob = 0

threadlock = threading.Lock()

def ioTask(cUrl):
    #read the website in as txt
    webFile = urllib.urlopen(cUrl)
    webString = webFile.read()
    #find the matching regex -> the numbers are always at the end.
    intString = webString[-3:]
    return int(intString)
    #return string -> num

def ioAdd(cUrl, num):
    #read the website in as txt
    altUrl = str(cUrl) + str(num)
    webFile = urllib.urlopen(altUrl)
    webString = webFile.read()
    #find the matching regex -> the numbers are always at the end.
    intString = webString[-3:]
    global glob
    glob += int(intString)
    #return string -> num

def ioAddLock(cUrl, num):
    #read the website in as txt
    altUrl = str(cUrl) + str(num)
    webFile = urllib.urlopen(altUrl)
    webString = webFile.read()
    #find the matching regex -> the numbers are always at the end.
    intString = webString[-3:]
    global glob
    global threadlock
    threadlock.acquire()
    glob += int(intString)
    threadlock.release()
```



```
49
50  def singleThreadUrlRead(iUrl):
51      #calls ioTask on 100 urls
52      #reset the glob to 0 for clearing previous calculation
53      glob = 0
54      for x in range(0, 100, 1):
55          altUrl = str(iUrl)+str(x)
56          global glob
57          glob += ioTask(altUrl)
58      #time the function for comparison. Sum of digits = 50470
59  startTime = time()
60  singleThreadUrlRead('http://lyle.smu.edu/~coyle/cse3342/testfiles/twestf')
61  endTime = time()
62  timeOne = endTime - startTime
63  print "Single Thread IO:  TOTAL = %d  TIME = %07f s" % (glob, timeOne)
64
65  def findNumAtUrlUpdateGlobalNoLocks(iUrl):
66      #Not Using locks, 100 threads to each read URL
67      global glob
68      glob = 0
69      threadList = []
70      for x in range(0, 100, 1):
71          threadList.append(threading.Thread(target=ioAdd, args=(iUrl, x)))
72      for y in threadList:
73          y.start()
74      for z in threadList:
75          z.join()
76      #Time ALL THE THREADS' time to finish
77  startTime = time()
78  findNumAtUrlUpdateGlobalNoLocks('http://lyle.smu.edu/~coyle/cse3342/testfiles/twestf')
79  endTime = time()
80  timeTwo = endTime - startTime
81  print "No Lock Multi Thread IO:  TOTAL = %d  TIME = %07f s" % (glob, timeTwo)
82
83  def findNumAtUrlUpdateGlobalWithLocks(iUrl):
84      #Not Using locks, 100 threads to each read URL
85      global glob
86      glob = 0
87      threadList = []
88      for x in range(0, 100, 1):
89          threadList.append(threading.Thread(target=ioAddLock, args=(iUrl, x)))
90      for y in threadList:
91          y.start()
92      for z in threadList:
93          z.join()
94      #Time ALL THE THREADS' time to finish
95  startTime = time()
96  findNumAtUrlUpdateGlobalWithLocks('http://lyle.smu.edu/~coyle/cse3342/testfiles/twestf')
97  endTime = time()
98  timeThree = endTime - startTime
99  print "With Lock Multi Thread IO:  TOTAL = %d  TIME = %07f s" % (glob, timeThree)
```

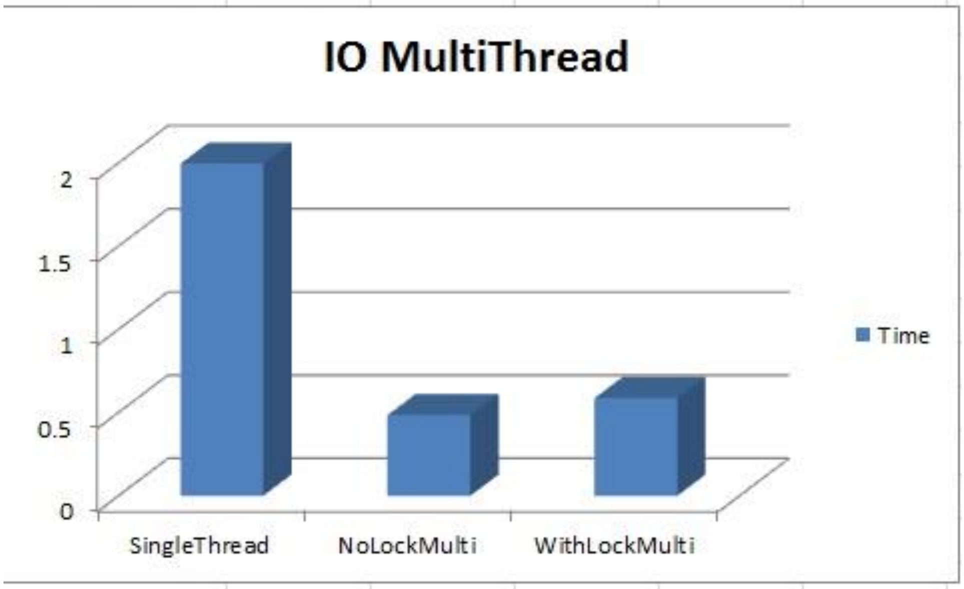
Result



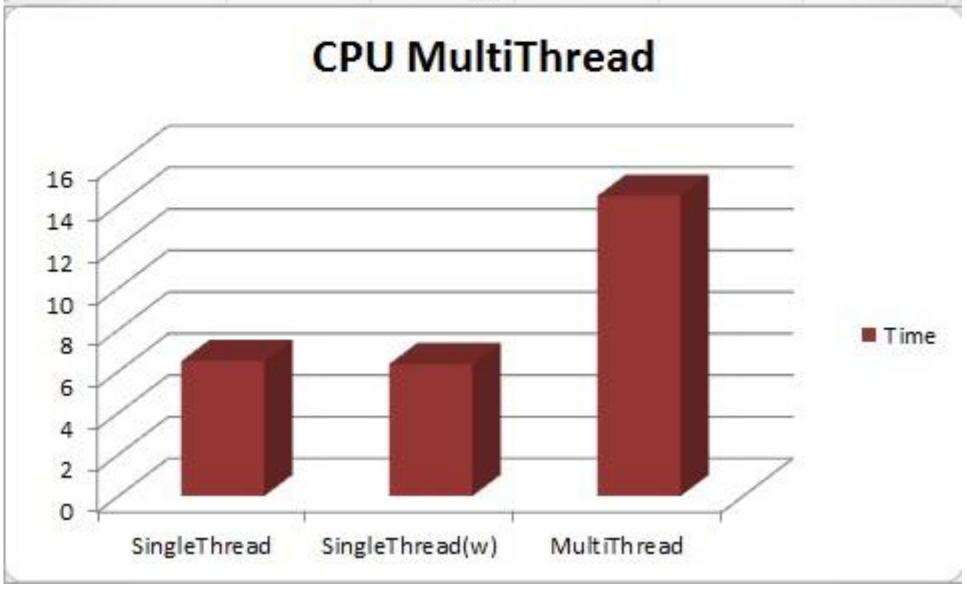
```
In [56]: runfile('C:/Users/lg/Desktop/hw10.py', wdir='C:/Users/
lg/Desktop')
C:/Users/lg/Desktop/hw10.py:61: SyntaxWarning: name 'glob' is
assigned to before global declaration
  threadList.append(threading.Thread(target=ioAdd, args=(iUrl)))
Single Thread IO:  TOTAL = 50470  TIME = 1.996143 s
No Lock Multi Thread IO:  TOTAL = 50470  TIME = 0.484653 s
With Lock Multi Thread IO:  TOTAL = 50470  TIME = 0.586618 s
```

Graph

	Time
SingleThread	1.996143
NoLockMulti	0.484653
WithLockMulti	0.586618



	Time
SingleThread	6.483771
SingleThread(w)	6.347731
MultiThread	14.43297



Analysis

For IO heavy task, the Thread and Lock only took 30% of single thread work

For CPU heavy task, the MultiThread actually 220% of a single thread

The cost of Lock is about 20% decrease in performance compared to without Lock Multithread

In Conclusion for IO intensive task Multithread is very helpful. In contrast, in CPU intensive task it can actually have worse performance.