Sorting It All Out

picture from http://en.wikipedia.org/wiki/Sorting

Objectives:

- To review more python
- To practice algorithm analysis

Background:

Once you have lists of items, one of the most common tasks is to sort the items, usually from smallest to largest. It is such a common task that the python list class has a sorting method included.

One thing of the main things we'll be doing in this class is to look beneath the magic that is python to see what is happening at a lower level. To that end, you should know that there are many, many sorting algorithms. We're going to try out a few of them. Here are the ones that you'll be doing:

- Python sort method
- Bubble sort
- Selection sort

In this lab, you will be testing them to see which is best. You'll need to write the python functions for bubble sort and selection sort based on their algorithms. Here's the pseudo-code from Wikipedia:
Selection sort (based on Wikipedia):

```
Procedure selectionSort(a : list of sortable items)
    for (j = 0; j < n-1; j++)
        iMin = j
        for ( i = j+1; i < n; i++)
            if (a[i] < a[iMin])
                iMin = i

        if ( iMin != j )
            swap(a[j], a[iMin])
```

Bubble sort (copied from Wikipedia):

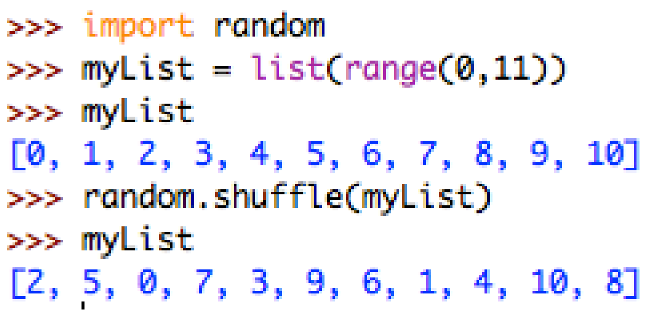
```
procedure bubbleSort( A : list of sortable items )
    n = length(A)
    repeat
       newn = 0
       for i = 1 to n-1 inclusive do
          if A[i-1] > A[i] then
             swap(A[i-1], A[i])
             newn = i
          end if
       end for
       n = newn
    until n = 0
end procedure
```

Assignment:

Your job is to figure out which sort is the best of the 3 given you (python sort, selection sort, and bubble sort) and one other sort. The other sort can be any sort that you find interesting. You may check your book or the internet for ideas for the other sort.

Write a driver file that will test all 4 sorting algorithms. Use a python list that contains random integers. Test each algorithm by using the following lists:

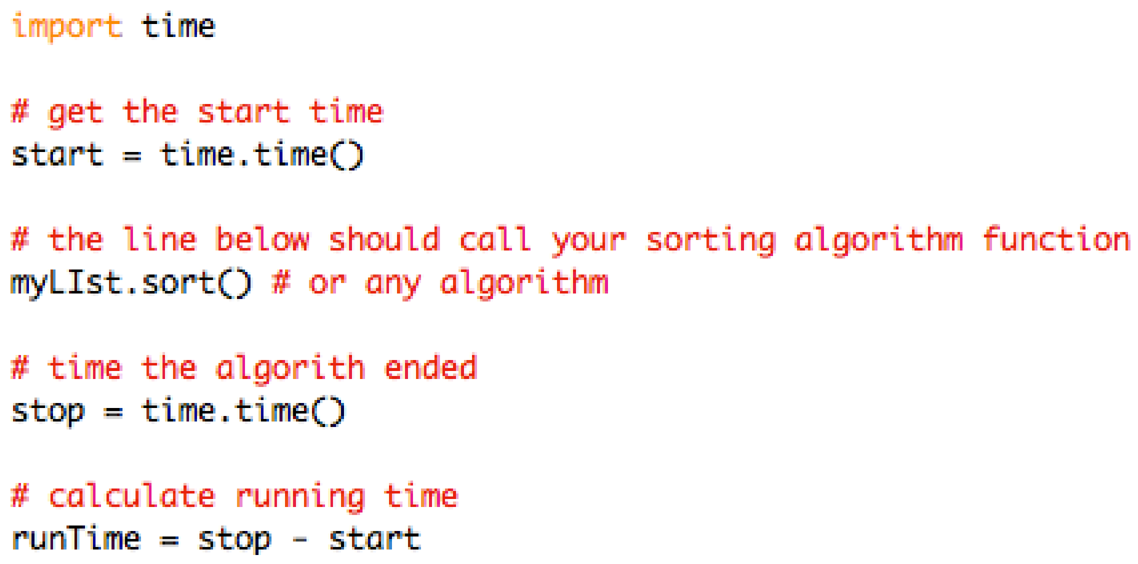
- List with 25 random(unsorted) integers from 1-25
- List with 100 random(unsorted) integers from 1-100
- List with 1000 random(unsorted) integers from 1-1000
- List with 10000 random(unsorted) integers from 1-10000



```
>>> import random
>>> myList = list(range(0,11))
>>> myList
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> random.shuffle(myList)
>>> myList
[2, 5, 0, 7, 3, 9, 6, 1, 4, 10, 8]
```

Figure 1: A simple way to create a unsorted list with unique numbers

Don't forget the timing function from chapter 3:



```
import time

# get the start time
start = time.time()

# the line below should call your sorting algorithm function
myLIst.sort() # or any algorithm

# time the algorith ended
stop = time.time()

# calculate running time
runTime = stop - start
```

There is only one demo for this lab. Show your completed program to your TA.

Program Requirements:

For the code:

- Since this is a paired assignment, your program should have the following comment block at the top.

```
#
#Names:        Name of one partner          Name of other partner
#Date Assigned:  Date of lab                Date Submitted:  Date submitted
#Course:       CSE 1384                     Lab Section:  Your lab section
#
#File name:  lastName.py  (of the person submitting file in myCourses)
#
#Program Description:  A short description of what the program is
#supposed to accomplish.
#
```

- Use appropriate comments throughout the code. Be sure to include comments before each class definition and before each method (member function) explaining the interface (what must be sent, what is returned) and the purpose of the method. Also, don't forget to make appropriate comments within the method definition.

- Make good use of whitespace. Be sure to include a blank line of space before each comment. Two lines of space between functions is nice, but one will suffice – whichever you choose, be consistent. Follow the Python programming style guide: https://www.python.org/dev/peps/pep-0008/#introduction

- Don't forget to use good **descriptive** variable names

For the lab report, follow the Lab Report Format Guide and complete the following sections:

- Title Page
- Analysis and Conclusions - The analysis section should include the bulleted information about your running environment and answers to the following questions:
  - Operating System of your computer
  - Processor
  - RAM
  - Plugged In / Battery Power ?
- Extra Analysis Questions:

  1. What is the time complexity of each of the 3 assigned algorithms and how does it relate to the results of your tests?
  2. What are the pros and cons of each of the 3 assigned algorithms?
  3. What other sort did you choose to test and why did you choose it?
  4. Find one more sort and explain how it works. What makes it different?
  5. What happens when you vary the length of the list being sorted? Why might you want to do that when testing an algorithm?

  If you use sources other than your text book, be sure to site your references.

Deliverables:

- Lab Report: Your lab report be completed after lab and printed out and turned in at the beginning of the next week's lab. It should follow the outline for this class (may be found on myCourses) and include:
  - Title Page
  - Test Plan (second paragraph from assignments)
  - Test Cases (run reach algorithm 3 times on all 4 algorithms and record the list size and runtimes)
  - Sample Execution
  - Analysis and Conclusions

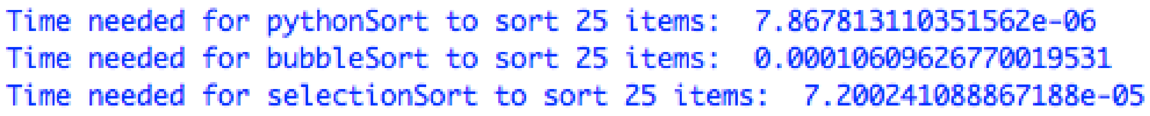
- Source Code: Be sure to submit the source code electronically in myCourses. You may submit the file more than once <u>before the deadline</u> if you find and correct an error after you submitted it the first time. You may not submit a correction after the deadline however.

Grading:

Task	Points
Demo	30 points
Lab Report: Analysis Questions	50 points
Rest of the lab report	20 points

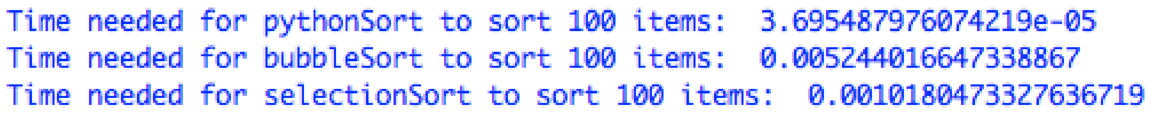
Sample Execution:

Run 1:



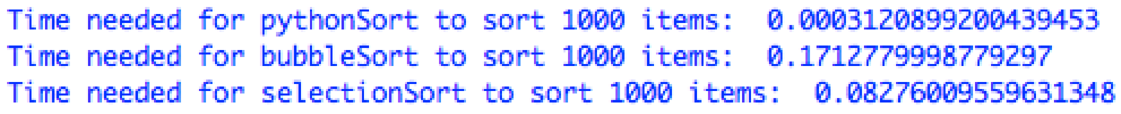
```
Time needed for pythonSort to sort 25 items:  7.867813110351562e-06
Time needed for bubbleSort to sort 25 items:  0.00010609626770019531
Time needed for selectionSort to sort 25 items:  7.200241088867188e-05
```

Run 2:



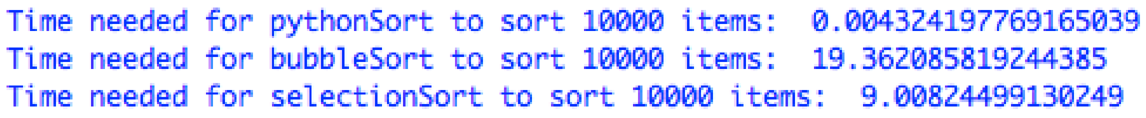
```
Time needed for pythonSort to sort 100 items:  3.695487976074219e-05
Time needed for bubbleSort to sort 100 items:  0.005244016647338867
Time needed for selectionSort to sort 100 items:  0.0010180473327636719
```

Run 3:



```
Time needed for pythonSort to sort 1000 items:  0.00031208992004394531
Time needed for bubbleSort to sort 1000 items:  0.1712779998779297
Time needed for selectionSort to sort 1000 items:  0.08276009559631348
```

Run 4:



```
Time needed for pythonSort to sort 10000 items:  0.004324197769165039
Time needed for bubbleSort to sort 10000 items:  19.362085819244385
Time needed for selectionSort to sort 10000 items:  9.00824499130249
```