9/20/13

4. Homework

Programming portion (problems 1 and 3) due **10/1/13** at 11:55pm on Blackboard.
Written portion (problem 2) due **10/2/13** at the beginning of class.

Please create a separate Python file for problem 1 and problem 3 below. Please use the following naming convention: `lastName_firstName_hw4_number.py` and submit it on Blackboard.

In order to receive any credit for the programming portions, you are required to thoroughly comment and test your code.

1. **Runtimes (7 points)**
   Consider the functions `my_min` and `my_min_slow` that we covered in class. The goal of this exercise is to compare the runtimes of both functions for lists of varying size.

   (a) (3 points) Write a program that creates lists of increasing sizes, runs both functions on those lists, and prints out the list size and the runtime for each list. This should result in a sequence of triples (list size, runtime for `my_min`, runtime for `my_min_slow`). You should have at least 10 such samples, and try to make the lists as large as possible.

   (b) (3 points) Produce a plot that shows both sequences of runtime data; you can use the plotting tool of your choice, possibly Excel. Which of these runtime functions grows faster, and why?

   (c) (1 point) Explain in words why `my_min_slow` correctly computes the minimum of the input array. What exactly do the for-loops do?
   In addition to the code, please create an electronic file with your answers to parts (b) and (c), including your plot, and upload the file to Blackboard.

2. **Code Tracing (6 points)**
   The goal of this exercise is to trace how variables change during the execution of code.

   For each of the code fragments below do the following: Trace the code, and for each time `#snapshot` is encountered, draw a picture of the current variable values in memory. Remember that the `#snapshot` comment inside the loops will be encountered multiple times, so you will have to draw the current variable values in memory for each of those encounters.

   (a)
```
x=0
i=0
while i<4:
  #snapshot
  x = x + i*i
  i = i+1

print x
#snapshot
```

Flip over to back page ⟹

(b)
```
x=1
list = range(1,5)
for i in list:
  #snapshot
  x = x*2

print x
#snapshot
```

3. **Pascal's triangle (9 points)**
The goal of this exercise is to write a program that prints Pascal's triangle:

```
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
[1, 5, 10, 10, 5, 1]
[1, 6, 15, 20, 15, 6, 1]
[1, 7, 21, 35, 35, 21, 7, 1]
```

Each row of Pascal's triangle has a `1` at the beginning, and a `1` in the end, and each other number is defined as the sum of the number left-diagonally above it and the number directly above it. So, the row `[1, 5, 10, 10, 5, 1]` is computed as `1`, then `5 = 1+4`, then `10 = 4+6`, then `10 = 6+4`, then `5 = 4+1`, and then `1`. The triangle above has 8 rows that are numbered $0 \ldots 7$.

(a) (2 points) As a warmup, write a function `fifth_row()` that computes the fifth row of Pascal's triangle from its fourth row. For this, you should assign `row = [1, 4, 6, 4, 1]`, and then write a loop that computes a new list called `newrow` from the numbers stored in `row`.

(b) (6 points) Write a function `pascal(n)` that prints rows `0` to `n` of Pascal's triangle. For this you will need two nested loops. The inner loop should look similar to the code for `fifth_row`. You may assume that $n \geq 2$. Test your function with several values of n.

(c) (1 point) What is the asymptotic running time of `pascal(n)` in terms of `n`? Please write your answer as a comment in your code, together with a very brief justification.

(d) **(Extra credit. This is not mandatory.)** Write code that prints Pascal's triangle in a neater layout as follows:

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5  10  10   5   1
    1   6  15  20  15   6   1
  1   7  21  35  35  21   7   1
```