Final Python Project

Sarah Mann

April 25, 2012

Seniors: Your assignment will be graded out of 35 points and is due Friday, April 27, 2012.
Not Seniors: Your assignment will be graded out of 100 points and is due Friday, May 18, 2012.

This document lists a number of problems which can be solved with the aide of a computer program. Your task is to solve some of them using Python. For each problem you solve, you will earn some points towards this assignment. The number of points earned for each problem is listed in the problem statement. The assignment will be graded out of 100 points; you may earn extra credit by accruing more than 100 points. For full credit on each problem you must write a Python program that

1. is your own original work.
2. is fully commented. Your comments must include
   (a) your name and the date
   (b) a statement of the problem being solved
   (c) a description of the approach you chose to solve the problem
   (d) documentation of the functionality of each portion of your code
3. works correctly on all test cases provided here.
4. work correctly on some other test cases similar to those listed here.

These problems were posed on the site `projecteuler.net`. Solutions to them can be found on the internet. Of course, use of work that is not your own is a violation of the honor code and will result in no credit for this assignment; don't look for code on the internet! You are welcome to discuss these problems with your classmates, but you may not share code.

Notes and Suggestions:

- For each problem, start by solving an easier related problem. Usually, this will mean using a small number for n or reproducing the example given.
- Generalize the procedure you used to solve the easier related problem. Write down a set of instruction (sometimes called "pseudo-code" or an algorithm) of how to solve this type of problem.

- Present your procedure to another classmate, Mr. Herzog, or Sarah. Can this third party follow your instructions to solve this problem correctly?
- Translate your procedure into Python code.
- Sometimes you may need to do things in Python that we have not yet specifically discussed. This is the nature of programming; you never know all the capabilities of a programming language and have to learn new tricks as the need arise. At the end of this document I have include a few new functions in Python that I think you might need. Mr. Herzog and Sarah can suggest other Python functions as needed.

Problem 1 (30 points)

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Write a program that asks the user for a value of n then computes the sum of all the multiples of 3 or 5 below n. Here are three examples:

- The sum of all the multiples of 3 or 5 below 10 is 23.
- The sum of all the multiples of 3 or 5 below 100 is 2318.
- The sum of all the multiples of 3 or 5 below 1500 is 524250.

Problem 2 (30 points)

Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be

$$1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

The sum of the even-valued terms in the Fibonacci sequence whose value does not exceed 50 is 44. Write a program that asks the user for a value of n, then computes the sum of the even-valued terms in the Fibonacci sequence whose value does not exceed n. Here are three examples:

- The sum of the even-valued terms in the Fibonacci sequence whose value does not exceed 50 is 44.
- The sum of the even-valued terms in the Fibonacci sequence whose value does not exceed 150 is 188.
- The sum of the even-valued terms in the Fibonacci sequence whose value does not exceed 15 million is 19,544,084.

Problem 3 (30 points)

The prime factors of 13,195 are 5, 7, 13, and 29. The largest of these is 29. Write a program that asks the user for a value of n, then computes the largest prime factor of n. Here are three examples:

- The largest prime factor of 13,195 is 29.
- The largest prime factor of 4,749,886 is 523.
- The largest prime factor of 59,331,030 is 7817.
- The largest prime factor of 155,131,173,990 is 2,381. (Note: if your program throws an error on this example, don't worry about it. It will not count against you.)

Problem 4 (30 points)

The sum of the squares of the first ten natural numbers is,

$$1^2 + 2^2 + ... + 10^2 = 385$$

The square of the sum of the first ten natural numbers is,

$$(1 + 2 + ... + 10)^2 = 55^2 = 3025$$

Hence the difference between the sum of the squares of the first ten natural numbers and the square of the sum is 3025 - 385 = 2640. Write a program that asks the user for a value of n, then computes the difference between the sum of the squares of the first n natural numbers and the square of the sum. Here are three examples:

- The difference between the sum of the squares of the first 10 natural numbers and the square of the sum is 2640.
- The difference between the sum of the squares of the first 50 natural numbers and the square of the sum is 1,582,700.
- The difference between the sum of the squares of the first 120 natural numbers and the square of the sum is 52,124,380.

Problem 5 (35 points)

By listing the first six prime numbers: 2, 3, 5, 7, 11, and 13, we can see that the 6th prime is 13. Write a program that asks the user for a value of n, then computes the nth prime number. Here are three examples:

- The 6th prime number is 13.
- The 100th prime number is 541.
- The 12,000th prime number is 128,189.

Problem 6 (40 points)

The sequence of triangle numbers is generated by adding the natural numbers. So the 7th triangle number would be $1 + 2 + 3 + 4 + 5 + 6 + 7 = 28$. The first ten triangle numbers would be:

$$1, 3, 6, 10, 15, 21, 28, 36, 45, 55, ...$$

Let us list the factors of the first seven triangle numbers:

triangle number	factors
1	1
3	1,3
6	1,2,3,6
10	1,2,5,10
15	1,3,5,15
21	1,3,7,21
28	1,2,4,7,14,28

We can see that 28 is the first triangle number to have over five divisors. Write a program that asks the user for a value of n, then computes the first triangle number to have over n divisors. Here are three examples:

- The first triangle number to have over 5 divisors is 28.
- The first triangle number to have over 10 divisors is 120.
- The first triangle number to have over 300 divisors is 2,162,160.

Problem 7 (45 points)

The following iterative sequence is defined for the set of positive integers:

$$n \to \begin{cases} \frac{n}{2}, & n \text{ even} \\ 3n+1, & n \text{ odd} \end{cases}$$

Using the rule above and starting with 13, we generate the following sequence:

$$13, 40, 20, 10, 5, 16, 8, 4, 2, 1$$

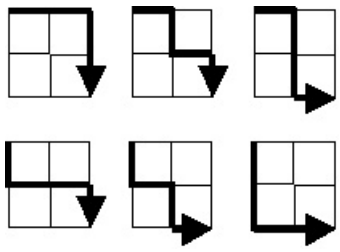
It can be seen that this sequence (starting at 13 and finishing at 1) contains 10 terms. Although it has not been proved yet (Collatz Problem), it is thought that all starting numbers finish at 1.

Write a program that asks the user for a value of n, then find the starting number less than n that produces the longest chain. NOTE: Once the chain starts the terms are allowed to go above n. Here are three examples:

- The longest sequence starting with a number less that 10 starts with 9 and has length 13.
- The longest sequence starting with a number less that 1,000 starts with 871 and has length 113.
- The longest sequence starting with a number less that 2 million starts with 1,723,519 and has length 349.

Problem 8 (45 points)

Starting in the top left corner of a 2×2 grid, there are 6 routes (without backtracking) to the bottom right corner.



Write a program that asks the user for a value of n, then finds the number of routes from the top left corner to the bottom right corner of and $n \times n$ grid without backtraking.

- There are 6 routes from the top left corner to the bottom right corner of a 2×2 grid.
- There are 184,756 routes from the top left corner to the bottom right corner of a 10×10 grid.
- There are 126,410,606,437,752 routes from the top left corner to the bottom right corner of a 25×25 grid.

Appendix: Useful Functions

This is a list of a few functions in Python you might find useful.

1. **The remainder operator** : $n \,\%\, m$ computes the remainder of n divided by m. If you type $8 \,\%\, 3$ into your IDLE, it will return 3 because the remainder of 8 divided by 3 is 2. You can test if an number n is even by checking if $n \,\%\, 2$ is zero. For example, $6 \,\%\, 2 = 0$ (since 6 is even), and $7 \,\%\, 2 = 1$ (since 7 is odd). How could you test if an number n is divisible by 3 using the % operator?
2. **The sum function**: `sum(LIST)` returns the sum of all of the items in `LIST`. For example, `sum([4,5,9])` returns 18. `sum(range(10))` returns $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 = 45$. Try these in your IDLE.
3. **floor and ceiling**: In the `math` library, the function `math.floor(x)` rounds x down to the nearest integer. For example, `math.floor(3.5)` returns 3.0. The function `math.ceil(x)` rounds x up to the nearest integer. For example, `math.ceil(3.5)` returns 4.0.