Homework 1

August 7, 2018

1 PHSX 331 Homework 1 - Practice with Python.

1.1 Due September 5th 5:00 p.m.

Goal of this assignment is to give you some practice doing simple tasks with Python. I would recommend doing these in order as each problem will get a little harder and you can use the previous problem to help solve the next one.

For problems 1 and 2 use the list foo,

```
In [3]: import numpy
        numpy.random.seed(0)
        foo = numpy.random.randint(0,100,25)
        print(foo)
```

```
[44 47 64 67 67  9 83 21 36 87 70 88 88 12 58 65 39 87 46 88 81 37 25 77
 72]
```

Problem 1: Suming a list / array **a)** Find the sum of all of the even numbers in the array "foo"

b) Find the sum of all of the odd numbers in the array "foo"

Problem 2: Equivalent resistance **a)** Write a function that takes a list of resistor values as an input and returns the equivalent resistance for all of the resistors being in serires. Check it for a circut with 2 Ω, 10 kΩ, and 143 Ω resistors

b) Now write a function that takes the same list of resistors and finds the equivalent resistance for the resistors when they are wired in parallel.

c) I want to write a program that could take in this picture of a simple circuit and find the equivalent resistance. Don't try to write a program that does this (unless you want to, but this a hard problem). Just think about how you would go about writiing a program do to this. What would the challenges be? What kind of things would you tell the computer to look at? Again I'm not looking for you to write a program for this, just brain storm ideas of how you might try to solve the problem.

Problem 3: Fun with Fibonacci **a)** The Fibonacci numbers are a set of numbers that are found by adding the last two numbers of the serires together. The first two numbers are given as,

```
In [2]: fib_num = [0, 1]
```

The next number in the series would be,

```
In [3]: fib_num[0] + fib_num[1]

Out[3]: 1
```

So the series would now look like,

```
In [4]: fib_num = [0,1,1]
```

Then we add the last two numbers of the series together, 1 + 1. So the fourth number in the series is 2. We can do this as many times as we want.

To make sure your code works test it to find all the fibinacci numbers less than 100. This way you can easily check that your code works by hand. The only input your function should have is the desired max number you want to find.

b) Find the sum of all even Fibonacci numbers less than 2 billion. If your program in question 6 is slow then this will take a really long time. Try to spend some time optimizing your program in question 6 first. If you can't optimize that's fine, you just might have to wait an hour to get the answer. (You should just combine your code from question 2 and 6)

For an extra challenge (you do not have to do this) see if you can do this without storing all of the numbers a list/array. We don't care about what all of the Fibonacci numbers are, we only care about the sum of the even numbers, so making a list of them and then adding up the numbers we care about is a waste of memory. It would be nice is we could forget about older Fibonacci numbers when we don't need them any more.

Problem 4: Prime or not Prime Write a function that tests if a number is prime or not. The function should take an input and give you back a True or False.

```
In [32]: prime_finder(3)

Out[32]: True

In [33]: prime_finder(10)

Out[33]: False

In [36]: prime_finder(732041)

Out[36]: True
```

Test your functino on the numbers 5, 231, 9851, 9853.

You can just brue force this. Take a number 'x' and divide it by all numbers less than it and see if any of these return a remander of zero. This will work for smaller numbers, but this will take a really long time for large numbers.

This is because you'll be double checking some numbers. When checking if 10 is prime or not, if you do 10/2 you then know that 5 will work as a divisor, so you don't need to tell the computer to check if 10/5 returns a remainder of zero.

Also you don't want to check higher numbers if you've already found an even divisor. When checking if 12 is prime, as soon as you know 12/2 gives an integer, you can stop checking larger numbers (look up what 'break' does in python).

Problem 5: Falling (with style) Write a function that takes a hight as an input (h) and finds the time it takes to hit the ground if dropped from that hight (t), assuming no air resistance, for the distances, 1 meter, 1,250 ft (Empire State Building), 254 miles (International Space Station). Be careful with your units.

1.1.1 Bonus Problem

(You don't have to do this! This is just an extra chalenging problem to get more practice if you want it. Don't feel bad if you don't do it.)

Bonus Problem 1 The fundamental theorem of arithmetic (FTA), says that all numbers greater than 1 are either prime, or can be writen as a product of prime numbers. Euclid proved this around 300 B.C. See Book VII, propositions 30, 31 and 32, and Book IX, proposition 14 of Euclid's Elements for proof.

Write a function that finds the prime factors for a given input number. As an example,

```
In [42]: prime_factors(1200)

Out[42]: [2, 2, 2, 2, 3, 5, 5]
```

Test this for 12, 1200, and 12345678987654321

When I test this with my code it takes,

```
In [49]: %timeit prime_factors(12345678987654321)

2.73 s ± 2.8 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

If you can beat my time I'll give you a point of extra creddit!