Problem Set 1

SFHS club whose name I still don't know!

October 17th 2016

1 Prime numbers

Eratosthenes (born ca. 276 BC in Cyrene, modern day Libya) was a Greek mathematician who proposed a method to identify prime numbers within a list of n natural numbers. The method consists in ruling out multiples of successive prime numbers until the list is exhausted. In the end, only the prime numbers remain. This method is now known as the Sieve of Eratosthenes.

Our goal now is to solve a slightly different problem: given a number n, what is the n-th prime number? (e.g., the 1st prime number is 2, the 2nd prime number is 3, and so forth)

In other words, the ultimate goal is to create a *function* **nth_prime** that accepts a natural number as an input and produces the n-th prime as the output:

```
>>> nth_prime(1)
2

>>> nth_prime(2)
3

>>> nth_prime(10)
29
```

Our strategy to solve this problem will be dividing it in smaller parts that are easier to solve and then putting it all together. You will find this strategy useful when solving several different problems in programming.

1.1 Is this number prime?

Create a function is_prime(n) that takes a natural number n as input and returns **True** if n is prime or **False** if n is not prime. The dirty way to do it is to create a for loop and check whether n is divisible by any number smaller than n (except for 1, of course).

In order to check whether a number a is divisible by a number b, you can use remainders (denoted by % symbols in Python) to find out if the remainder in the division a/b is zero:

```
>>> 10%3   # Not divisible
1

>>> 10%4   # Not divisible
2

>>> 10%5   # Divisible
0
```

Another tool you will find useful is the **if** statement. The if statement works by first evaluating a header. If the condition described in the header is met, the Python interpreter will proceed to the first line inside the if statement. If the condition in the header is not met, the if statement is skipped altogether:

```
n = 99
if n == 99:    # Will proceed into the if statement if n is equal to 99
    print('n is equal to 99')

if n != 99:   # Would proceed into the if statement if n were different than 99
    print('n is different than 99')

if 99%11 == 0:   # Will proceed into the if statement if the remainder of 99/11 is 0
    print('99 is divisible by 11')

if 99%5 != 0:   # Will proceed into the if statement if the remainder of 99/5 is not 0
    print('99 is not divisible by 5')
```

1.2 Find the n-th prime

Create a prime number counter prime_counter that starts at 0. Using your new is_prime() function, iterate over the natural numbers and add 1 to prime_counter every time a new prime number is found. Do this until prime_counter reaches the desired n, then stop iterating.

You will find the **while** statement useful in this case. The while statement works by evaluating a header and repeating the contents of the statement while the conditions in the header are still met (in other words, until the conditions in the header are not met anymore):

```
a = 0
while a < 5    # Adds 1 to the variable a while it is smaller than 5
    a = a + 1

print(a)    # This should return 5
```

1.3 Play around with your new function

What is the 23rd prime number? Find the answer and verify that it is correct using an independent reference (such as an online list of prime numbers).

Observe how much time it takes for your function to find the answer. What is the largest prime number you can find in less than one minute of processing time? In 5 minutes? Can your program find the 10000th prime number in less than 15 minutes?

2 Drawing a picture

The 'turtle' we went through during the meeting this week is actually a pretty useful tool for drawing a picture. Here I propose the challenge of drawing a picture with the name of your school "SFHS" in it. Besides that, you can draw whatever you like to make it beautiful. You can also try to draw your name in the same picture if you are interested!