Homework 3

Due Monday, October 21, 2019

Math 206

1. Last week we wrote pseudo-code for the Sieve of Eratosthenes: To find all the primes ≤ 30,

   (a) Create a list `isPrime` consisting of True 31 times. We want 31 elements rather than 30 because we're going to ignore `isPrime[0]`.

   (b) Set `isPrime[1] = False`.

   (c) For i for 2 to 30, if `isPrime[i]` is true, then for $j = 2i, 3i, 4i, \ldots$ set `isPrime[j] = False`.

   We said that the following code would be helpful to start:

```
isPrime = [True]*31
isPrime[1] = False
for i in range(2,31):
  ...
```

   Finish implementing the sieve, and make sure that it gives sensible answers up to 30. Then rewrite your code to find all the primes up to 1 million.

   You might first experiment with the following code, which says "print all integers k, starting from $k = 14$, incrementing by 7 (or 8) each time, and quitting when you're ≥ 31.

```
for k in range(14,31,7):
  print(k)

for k in range(14,31,8):
  print(k)
```

2. The first prime is 2, the second is 3, and the third is 5. Find the 10,001st prime. Hint: It's less than 1 million.

3. Find all the prime factors of 600851475143 that are less than 1 million. Is that all of its prime factors?

4. "Twin primes" are pairs of prime numbers that are two apart: for example 11 and 13, or 17 and 19, or 29 and 31. It is conjectured that there are infinitely many pairs of twin primes. Find all twin primes ≤ 1 million.

5. For a randomly chosen interval of 14 consecutive integers under 1 million, what is the probability that the interval contains at least one prime? Write some code to approximate this.

   It may be helpful to know that the following code gives a random integer between 1 and 1 million.

   ```
   import random
   N = random.randrange(1,1000001)
   print(N)
   ```

6. Prime numbers of the form $2^n - 1$ are called *Mersenne primes*. It is conjectured that there are infinitely many Mersenne primes. Find all the Mersenne primes less than 1 million.

7. (a) Write a function `smallest_prime_divisor` that takes an integer $n \leq 1$ trillion and returns its smallest prime divisor. Note that you only have to check divisors up to 1 million. Check that it gives sensible output.

   (b) Use your function to investigate the smallest prime divisors of numbers of the form $2^n + 1$, where $1 \leq n \leq 39$. I suggest writing a program that prints n and the smallest prime divisor of $2^n + 1$ side-by-side. Note that `print(a,b)` will print two numbers `a` and `b` side-by-side.

   (c) Do you see any patterns in your list? Try to precisely and mathematically staet at least three patterns that you find. Some possiblities here might look like "If n is ____ then then $2^n + 1$ is ____.

   (d) See if your patterns continue for bigger primes.