Problem I.8

Primality Function

Due Date: 2/22/2019
Folder: `IntroToProgramming`
File Name: `I8_PrimeFunc_LastName.py`

Learning Objectives

- Programming Skills
  - ⋆ Functions
  - ⋆ Conditionals
  - ⋆ Loops
- How to test for primality

Problem Background

The basic building blocks of integers are the *prime* numbers. Prime numbers are numbers that are divisible only by divided by themselves and 1. The Fundamental Theorem of Arithmetic says that every natural number can be written uniquely as a product of prime numbers. This makes the prime numbers like the atoms of natural numbers, combining them in different ways gives you a different natural number. Therefore, it is often very nice to test whether a particular number is prime or not. Note, if a number is not prime, it is called *composite*.

In this problem, you will write a function to test whether a particular number is prime. This function will be useful to keep around, as this test is used in multiple other projects in the future. One interesting fact that might be helpful in writing this function; if a number n is composite, the largest prime divisor is no bigger than $\sqrt{n}$.

Rules for this project: There are many algorithms that test the primality of a number, and I'm sure all of them could be found online very quickly. You are not allowed to look online for these algorithms. I want you to come up with your own algorithms to check if a number is prime. We might consider some better algorithms in the future, but right now I want to see what you come up with using your own creativity and skills.

Programming Reminders

- Syntax for a function: `def func_name(param1,param2,...):`
- Loops keyword: `for x in range(N)`, `while(condition)`

Program Criteria

Write a program that does the following:

- Define a function called `prime_check` (with `def`) that takes as input a natural number, and returns `True` if that input is prime and `False` if that input is composite.
- Create an input variable `n`, determining the largest prime number you will find below.
- Use your `prime_check` function to determine the $\mathtt{n}^{th}$ prime number, and print it out with appropriate descriptive text. (Note: 1 does not count as a prime number, so for instance the 1^{st} prime number is 2, and the 2^{nd} prime number is 3.)

Deliverables

Place the following in a folder named `IntroToProgramming` in your repository:

- A Python file `I8_PrimeFunc_LastName.py` that satisfies the program criteria.
- A PDF file written with Latex named `I8_PrimeFunc_LastName.pdf` that describes how your primality test algorithm works.