2022 Spring -- CSCI 1300L
Project 01: Prime Number Checker

Introduction

This project is used to conclude what we have learned so far in Python programming. Topics include the use of variables, data types, basic computations, input/output processing, programming logic, design of a program, decision structures, control structures, and so on. By completing this project, it is expected that you can further understand and apply basic software development process in making a computer program.

Project description

A prime number is a natural number greater than 1 that is not a product of two smaller natural numbers (see source: https://en.wikipedia.org/wiki/Prime_number). Our "Prime Number Checker" program works in the following way:

A user inputs a positive whole number between 1 and 500. The program evaluates the input and determine if such a number is a prime number. After determination, the program will output the corresponding message whether the input is a prime number or if such a number is divisible by a smaller positive number other than 1, i.e. a composite number.

Project in detail:

While grading for this project is based on the Python code only, we suggest that you use a software development process, such as requirement->analysis->design->implementation->testing, to make the program accurately and effectively.

The general requirement is outlined in the project description. By understanding the requirement, you will analyze the problem: what functions (e.g. input and output) will be needed, what conditions must be met, whether there is any other need that should be implemented, etc.

Followed by the analysis, you will focus on the design of the program. We outline one program design in this project document. However, you may discover or think of a better design. Feel free to use your own design to make the program implementation.

Program design:
(1) The program allows the user to input a value (X). Note that the user input may be anything. However, only an integer from 1 to 500 (both inclusive) is recognized as a valid input. Think how you should handle the user input.

(2) Assuming that the user input is valid, the program should then evaluate if such an integer is divisible by some numbers.

- If the input value X is 1, the program should output a message such as "1 is neither a prime or composite number."
- If the input value X is not 1, the program will try X to be divided by a smaller number Y, where Y is between 1 and X. For example, if the input value X is 60, such a smaller number Y can be any such as 2, 7, 55, ... By doing this action, the program tries to find a divisor of that integer X.

Obviously, if division result is an integer, such a number Y is a divisor of X. Think how to check if a number is divisible by another number, as there are several different ways.

The program will iterate the division process by choosing another Y and determine if such a new Y is a divisor of X. The iteration can be processed from the smallest Y to the largest Y (*: see the note below). For example, if X is 11, the program will try to do: 11 divided by 1, 11 divided by 2, 11 divided by 3, ... Think how to implement the code to complete the iteration process. During the iteration, keep the count N of the calculated result to see how many divisors (N) there are for such an integer X.

- If the count N is 2, i.e. such a number X is divisible by 1 and by itself, then X is a prime number by the mathematical definition. For example, 17 is divisible by 1 and 17 only (two divisors). Therefore, 17 is a prime number.
- If the count N is greater than 2, i.e. such a number X is divisible by 1, by itself, and by at least another integer that is smaller than X, then X is a composite number. For example, 9 is divisible by 1, 3, and 9 (three divisors). Therefore 9 is a composite number.

(3) Print the output based on the calculated result, either X "is a prime number" or "is a composite number".

After you understand the program logic as specified above, you may write the pseudo code first and then convert the pseudo code into the Python code.

When the Python code is written completely, do test your program thoroughly. Try to input different values and check if the program will output the correct result. Keep in mind, thoroughly testing your program is critically important. A robust program should work correctly and efficiently in all possible scenarios. Consider to write code comments appropriately so that both you and anybody else who sees the code can better understand how the code works.

Note on (*): Think how to improve the efficiency of the program. Do you really need to do 60 iterations for checking whether the number 60 is a prime number? Try to revise the design and the code to make your program work better.

<u>You absolutely CANNOT use or modify any existing code of "primality test" or any similar program, which can be found from any textbook or online, to complete this project assignment. You must write the program line by line by yourself.</u>

Your program's input or output does not need to be exactly the same as what is specified in the program design. However, your program must:
(1) handle the user input of a valid number successfully,
(2) give a correct output based on calculation, and
(3) COMPUTE by applying a sound logic and necessary calculations. (#: see the note below)

Note on (#): For example, you cannot write a so-called "program" by listing all known prime numbers and just tell "hey, this is a prime number because it is in my list"…In such a case, your program does not compute.

Submission instruction

After you have completed the project, upload and submit the Python source code file *Project01.py* to eLC. Always double check that your submission was successful on eLC.

Grading

A score between 0 and 5 will be assigned.

1. The program can successfully process a user's valid input (0.5) and invalid input (0.5), including non-integers or a number outside the specified range. (1 point)
2. The program should pass the test by correctly processing number 1 (1 point), at least 2 prime numbers (1 point), and at least 2 composite numbers (1 point). The prime numbers and composite numbers to be used to test your program will be randomly selected by the grader. (3 points)
3. Only a single source code file is submitted and no other file is submitted (0.5) and the entire Python program can be executed without any additional error (0.5). (1 point)

Special notice regarding the submission:

Late submission penalty. Points will be deducted from the original grade. If your submission is after the posted deadline…

(1) within 24 hours: -2
(2) between 24 hours and 48 hours: -3
(3) between 48 hours and 72 hours: -4
(4) after 72 hours: project will not be accepted.