GEORGE MASON UNIVERSITY
THE VOLGENAU SCHOOL OF ENGINEERING

CYSE/SYST 130
INTRODUCTION TO COMPUTING FOR DIGITAL SYSTEMS ENGINEERING

Project No. 1

A. K. Zaidi

Due: Oct 22, 2021

Purpose: The purpose of this assignment is to provide you practice in writing basic Python programs and practice basic Python control structures, loops, and I/O functions. You may require built in data structures of Lists, Dictionary and/or Sets for this assignment.

- Honor Code: This is an individual assignment. You can ask the TA and Instructor if you have questions regarding the homework.
- Syntactically Correct Programs: You need to turn in syntactically correct and working programs. No partial credit will be given for programs that are not syntactically correct.
- Coding Style Guidelines: Please follow the coding guidelines provided in the link below while writing Python code. It is a coding best practice to stick to coding guidelines and write well documented and consistent code. https://www.python.org/dev/peps/pep-0008/

Submission Guidelines: Please do the following to submit your solutions:

- Homework Guidelines: Please start your work early as it takes time to understand and debug programs. Homework assignments cannot be completed in a single sitting and would require you to come back to the problems a few times with breaks in between.
- Homework Deadlines: You need to complete your work on time. No partial credit will be given for work submitted after the due date.
- Homework Work Package: Please keep all your programs and data in a single directory and following the naming convention First_Last.zip.
- Solution Notebooks: You are encouraged to submit your solutions electronically as Juypter Notebook. The notebook should be named as Project1_First_Last .ipynb.
- TA Instructions: TA may announce additional/modified instructions for submission. It is imperative that you should follow TA-issued instructions.

You may share ideas, links to electronic resources, and discuss the project tasks with other students. However, you should not share your code with others. Sharing and/or exchanging your code partially or in its entirety will be considered a violation of the honor code.

Task 1:

A primality test, as described by Wikipedia, is an algorithm for determining whether an input number is prime. Among other fields of mathematics, it is used for cryptography. Another important problem in cryptography is prime factorization – a computationally difficult problem, whereas primality testing is comparatively easy (its running time is polynomial in the size of the input).

Wikipedia also defines:
A prime number (or a prime) is a natural number greater than 1 that is not a product of two smaller natural numbers. A natural number greater than 1 that is not prime is called a composite number.

The following are four methods one can find in textbooks and online sites on how to implement a primality test in a given programming language:

Method #1: Given an input number, n, check whether it is evenly divisible by **any number between 2 and n** (i.e. that the division leaves no remainder). If so, then n is composite. Otherwise, it is prime.

Method #2: Given an input number, n, check whether it is evenly divisible by **any number between 2 and n/2** (i.e. that the division leaves no remainder). If so, then n is composite. Otherwise, it is prime.

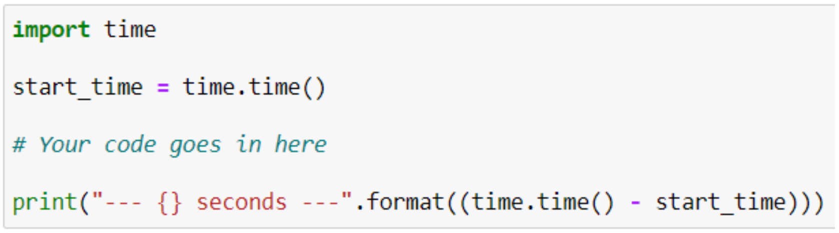
Method #3: Given an input number, n, check whether it is evenly divisible by **any number between 2 and √n** (i.e. that the division leaves no remainder). If so, then n is composite. Otherwise, it is prime.

Method #4: Given an input number, n, check whether it is evenly divisible by **any <u>prime</u> number between 2 and √n** (i.e. that the division leaves no remainder). If so, then n is composite. Otherwise, it is prime.

In this task, you are asked to implement the four methods above in Python as separate functions. Use all four functions to calculate all prime numbers between 1 and 100 and report the output from each.

In the second part, you are asked to give your reasons for choosing one of the four algorithms over others especially when you are required to run primality test on very large numbers. Give arguments both in favor and against each algorithm.

You may find the following code template useful in case you like to observe the time it takes for a function or a code script to complete its execution:



```python
import time

start_time = time.time()

# Your code goes in here

print("--- {} seconds ---".format((time.time() - start_time)))
```

Task 2:

A course instructor has recorded the homework assignment, midterm, and final examination grades in an Excel worksheet. The file CYST003.xls with this information is provided to you as an attachment.

The grading distribution of this course is as follows:

- Homework Assignments are 60% of grade
- Midterm is 20% of grade
- Final Exam is 20% of grade

Letter grades are decided as follows:

94% and above	A	73-75%	C
90-93%	A-	70-72%	C-
86-89%	B+	66-69%	D+
83-85%	B	63-65%	D
80-82%	B-	60-62%	D-
76-79%	C+	at or below 59%	F

In this task, you are asked to develop a Python program that reads the student grade information from the Excel file, calculates the aggregate final score using the grading structure above, and then assigns each student a letter grade using the grading ranges given in the table above. It should print the results of these calculations in the form: `<first_name  last_name: score=value, grade=value>`

E.g., `Nicolas Cage: score = 98, grade = A`

Extra Credit: You may earn extra credit if you can also update the Excel file by adding two more columns (i.e., **score** and **grade**) to it and writing the calculated values for each student in the corresponding row.

Resources on reading/writing from/to Excel files using Python:
https://www.geeksforgeeks.org/reading-excel-file-using-python/
https://www.geeksforgeeks.org/writing-excel-sheet-using-python/