Introduction to Python,
Prof. Dr. David B. Blumenthal
Suryadipto Sarkar

Python Student Projects

April 11, 2022

Instructions

Choose one of the following topics as your final student project. You will be graded on the execution of the sub-parts specified in each of the topics. Marks will be awarded for partial implementation. You are welcome to refer to online resources, but please refrain from blatant plagiarism. Happy coding!

Project-1: Roman numeral converter

Write a python program that takes as input a Roman numeral, and returns its corresponding decimal. As test cases, convert the following Roman numerals to their decimal:

1. MXIV

2. XXXVII

3. MDCLXIII

Project-2: Metric/ unit converter

Different parts of the world follow different measurement units. Write a program to convert the following measurements to the international (SI) units (Don't hard-code it; implement it such that the user can input a value and get its SI unit as output.):

1. A car in Arizona runs at 52 km/hr. Display its speed in m/sec.

1. A man in Wales weights 2 stones. What is his weight in kg?

2. A flight from Phoenix to Dayton allows 24 ounces of liquid in the carry-on. How much is that in kg?

Project-3: Single-player game (rock-paper-scissors/ dice rolling game)

Write code to implement a computer vs. single-player game of rock-paper-scissors/ roll the die; and display both the computer and user scores after three attempts. (For dice rolling game, higher total score wins.)

Project-4: Password strength checker + random password generator

Write code to implement a program that asks user to input a password, then does the following:

1. If password contains atleast one small alphabet, one capital alphabet, one number, one '@' and one '#', and is more than 12 characters in length: ***Display 'Strong password' and end program.***

2. Else: ***Display 'Weak password, try again!' and accept another input. After a total of three failed inputs, display a randomly generated password that fulfils all of the criteria. End the program.***

Project-5: Number guessing game

Write code to implement a program that does the following:

Step-1: generate (but do not display) a random number between 1 and 100.

Step-2: Display 'Enter any whole number between 1 and 100. Enter STOP to end the game.'

Step-3: Accept input from user

Step-4: If number input is higher, display 'Too high.' If input is lower, display 'Too low.' If input is correct, display 'Correct!' Continue for as long as user either gets the number right, or types 'STOP' to immediately terminate the program.

Project-6: Social media censorship app

Many apps and websites (most notably, Youtube, Twitter and Facebook) actively censor articles containing fake news, or controversial political views. Of course, they make use of Natural Language Processing (NLP) for this purpose. But we will try implementing a simplified version of a language censorship tool:

1. Define an array of 50 flag keywords. (Example: 2020 elections, voter fraud, Donald Trump, Joe Biden, parliament, fake news et cetera.)

2. Assign a random score between (0,1) to each of these keywords.

3. Ask user to enter a paragraph of text containing atleast 20 and atmost 300 words.

4. Generate a final flag score as per the total sum of scores associated with appearing flag keywords in the paragraph.

5. If score is greater than 15, print 'Your article will be taken down.' Else, print 'Article uploaded successfully!'

Project-7: Exercise Tracker App

Write code to implement a gym exercise tracking application that does the following: 1. There are three categories of exercises: Free weight exercises, machine exercises, cardio. 2. There

is a daily quota of 1000 points that the user has to achieve to win a 'Gold Star' rating. 3. Each of the free weight exercises contain 200 points per 10 reps; each of the machine exercises contains 150 points per 10 reps; each of the cardio exercises contains 50 points per minute. 4. Ask user to choose exercise type, and log in the number of reps (or minutes). And based on that display the points that the user has scored thus far. 5. Note that no user should be allowed to score more than 400 points per exercise category (that is, every user must have to score a minimum of 200 points per category). 6. The program ends only when the user enters 'STOP'. Finally, the category-wise and total scores are displayed.

Project-8: Mad libs game

Write code to implement a mad libs game that asks user for the following inputs:

1. **Enter a common noun**

2. **Enter a verb**

3. **Enter the name of a place**

4. **Enter a nonsense word**

5. **Enter the name of a fictional character**

6. **Enter another verb ending with 'ing'**

Once the user has input all these words, generate a mad lib sentence (that does not necessarily have to be grammatically correct), and display it. Repeat this for three times (both reading input and displaying the generated sentence). Automatically end the program after the third time; or if the user types in 'STOP'.

Note: You need to include the actual input in the generated sentence. However, the rest of the sentence can be hard-coded.

Project-9: Who wants to be a millionaire?

Write code to implement a multiple-choice quiz that does the following:

1. A total of 10 questions, each worth $100,000.00. After each correct question, the user is asked if he wants to continue playing or wants to walk away with his current winnings. An incorrect answer means he loses everything. But he cannot quit with his winnings after a question has been displayed.

2. He is given only one chance in the entire game to change to a different question. But, he cannot change questions on the last question. 3. At the end of the game, display his total winnings.

Project-10: File reading and simple text analysis

Language processing is a very interesting and prospective domain. However, we will not delve into the learning aspects of language processing, in the interest of time and effort required. In this exercise, you are supposed to read a text file named ***All Along the Watchtower.txt***, and display the top ten most occurring words. Subsequently, you must draw a bar plot where the x-axis label are the words themselves, and the y-axis labels are how many times each of these

words appears in the text. (You can optionally make use of the ***matplotlib*** library for generating the plot.)

Note: The text file 'All Along the Watchtower.txt' has been included along with the assignment package.
