COMP 1405B

Fall 2019 – Practice Problems #3

Objectives

- Continue to practice using basic Python inpute, output, and calculations
- Practice writing code with basic branches
- Practice writing code with nested branches
- Practice writing code with flattened if-else if-else branches
- Practice using logical operators within if/then statements
- Practice applying problem solving and computational thinking skills to programming problems

Remember to test your solutions to verify their correctness and discuss your solutions with both other students and TAs in the course. Verifying your solutions with TAs will be a good way of ensuring that you are on a path to success in the course.

Problem 1 (Simple Calculator - *)

Write a program that performs the function of a simple integer calculator. This program should ask the user for an operation, including the functionality for at least addition, subtraction, multiplication, and long division. After the user has selected a valid operation, ask the user for the two operands (i.e., the integers used in the operation). If the user enters an invalid operation, the program should print out a message telling them so.

Please also note that long division of integers requires that you provide both a quotient and a remainder - refer to https://docs.python.org/3/library/stdtypes.html - numeric-types-int-float-complex if you do not recall how to compute the remainder when you divide two integers.

Sample Outputs (user input highlighted)

```
(A)ddition
(S)ubtraction
(M)ultiplication
(D)ivision (Long)
Please select an operation from the list above: Vaporize
This program does not support the operation "Vaporize".
```

```
(A)ddition
(S)ubtraction
(M)ultiplication
(D)ivision (Long)
Please select an operation from the list above: M
Please provide the 1st integer: 3
Please provide the 2nd integer: 5

3 * 5 = 15

(A)ddition
(S)ubtraction
(M)ultiplication
(D)ivision (Long)
Please select an operation from the list above: D
Please provide the 1st integer: 16
Please provide the 2nd integer: 3

16 / 3 = 5 with remainder 1
```

Problem 2 (Leap Years - **)

A leap year is a year that has 366 days instead of the normal 365. Leap years occur every 4 years (e.g., 1896, 2020), *except* every 100th year is not a leap year (e.g., 1900). However, every 400th year, the 100th year rule is ignored and the year is a leap year (e.g., 2000).

Some examples:

```
1896 is a leap year
1996 is a leap year
1900 is not a leap year
2000 is a leap year
2018 is not a leap year
2020 is a leap year
```

Write a program that asks the user for a year, and outputs whether or not it is a leap year. Hint: this problem is all about figuring out what numbers are divisors of the current year (4, 100, 400) and using if/elif/else statements to determine if it is a leap year or not.

Problem 3 (Movie Expert System - *)

Write a program that presents the user with four possible movies that an actor/actress has been in. The user should mentally select one of these movies. The program should then ask the user exactly two questions about the content of the movie they picked and, using the user's responses, determine which of the four possible movies was being considered by the user. To accomplish this your program will contain three calls to the input() function but only two of them will ever actually occur (i.e., the decision to ask the second or third question will depend on the response you receive for the first question).

This is known as an expert system and for this exercise you should restrict your expert system to movies starring a specific actor of your choice. Consider, as a clarifying example, the spoiler-filled sample output below for an expert system associated with Nicolas Cage:

Sample Output (1 of 2)

```
The actor being considered is Nicolas Cage.
The possible movies are: "The Wicker Man", "Next", "Kick-Ass", and "Con Air".

Does his character die by the end of the movie and stay dead? Yes
Does his character ever use the phrase "Not the bees!"? Yes

The movie you are thinking of is "The Wicker Man".
```

Sample Output (2 of 2)

```
The actor being considered is Nicolas Cage.
The possible movies are: "The Wicker Man", "Next", "Kick-Ass", and "Con Air"

Does his character die by the end of the movie and stay dead? No
Can his character see into the future? No

The movie you are thinking of is "Con Air".
```

Problem 4 (More Currency Conversion - **)

Write a program that will allow the user to choose two currencies out of a set of available currencies to convert between. For example, your program may support the following currencies: CAD, USD, GBP, EUR, and JPY (you can add as many as you want). Print out a menu allowing the user to specify which currency they are starting with and what currency they want to convert to. The program should then ask the user how much currency they are converting, perform the calculation, and print out the resulting amount of the new currency. You can specify the conversion ratios for each pair of currency directly in your code and they do not need to be accurate to the real-world currencies. Try minimizing the amount of code that you must write for this problem. If you approach this without planning, you may end up with many nested branches and confusing code.

Problem 4 (Car Stop/Go Version 1 - *)

For this question, you will write a program that determine if a car should go through an intersection or stop. Your program will prompt the user for the colour of the traffic light at the intersection (a string), the distance to the intersection in meters (a float), and the speed of the car in meters per second (a float). Your program must then determine if the car should go through the intersection or stop. As driverless cars are not included in the School's budget (yet), it is suitable for you to just print out the action that the car should take.

The following rules should govern the car's decision:

1. "Go" when the light is "green"
2. When the light is "yellow", "Go" if the car will reach the intersection within 5 seconds. Otherwise, the car must "Stop"
3. When the light is "red", "Go" if the car will reach the intersection in 2 seconds. Otherwise, the car must "Stop"
4. If the light is any color other than "green", "red", or "yellow", the car must "Stop"

Problem 6 (Car Stop/Go Version 2 - *)

In Python, you can combine multiple Boolean statements together to form a single complex Boolean expression. To combine multiple clauses that must all be true, you can use the keyword 'and'. To combine multiples clauses where at least one must be true, you can use the keyword 'or'. As examples, we could use either of the code blocks below to enforce a double pass rule in a course where the student is required to pass the coursework (assignments, etc.) and the exam in order to pass the course:

```
if course_grade >= 50 and exam_grade >= 50:
      print("You pass the course!")
else:
      print("Sorry, you failed the course.")

if course_grade < 50 or exam_grade < 50:
      print("Sorry, you failed the course.")
else:
      print("You pass the course!")
```

Implement a program that produces the same output as the previous problem using only a single if/else (i.e., no nested if structures, no elif blocks).

Problem 8 (Trivia Game Progressive Problem 1 - *)

This problem is part of a set of progressive problems that will aim to build up a complex system by the end of the course. If you work through the "Trivia Game" progressive problems throughout the course, by the end of the term you will have created a trivia game that retrieves questions from the Internet, keeps track of a user's score throughout a game, and remembers the high scores forever.

We will start with a simplified version of a trivia game (decomposition!) and continue to add details throughout the term. For now, write a program that prints out a single trivia question that you made up, gets an answer from the user, and then tells them if they were correct or not.

One thing you may want to consider is case sensitivity. For example, "Ottawa" is not considered equal to "ottawa" within a programming context. We will talk about handling string data in more detail later, but for now all you need to know is that Python has a way to convert strings to lower/upper case. If x is a string variable, then x.upper() evaluates to the uppercase version of x. There is also x.lower(). So if x = "Ottawa" and y = "otTAwa", x does not equal y, but x.lower() equals y.lower(). Like the int/float/str type conversions, this does not actually change the value of x but only evaluates to the lowercase representation of x.

If you wish to extend this problem now, you can consider adding multiple questions to the trivia game. You can keep a score of the user's correct/incorrect answers and give them a grade once all questions have been asked.

Problem 8 (Unbeatable Tic-Tac-Toe AI Progressive Problem 1- *)

This problem is part of a set of progressive problems that will aim to build up a complex system by the end of the course. If you work through the "Unstoppable Tic-Tac-Toe AI" progressive problems throughout the course, by the end of the term you will have created a text-based tic-tac-toe game that allows a human user to play against an unbeatable AI player. The AI algorithm that we will apply within these problems near the end of the course will be generalizable to other games similar to tic-tac-toe too.

To start, we will just work on using if statements to determine if a player has won a tic-tac-toe game or not. If you are unfamiliar with the rules of tic-tac-toe, you can read about the game here: https://en.wikipedia.org/wiki/Tic-tac-toe. To start, download the

PP3-Resources.zip file from cuLearn, extract the contents, and open the tic-tac-toe.py file.

Within this code file, 9 variables are defined, each of which has two letters in the variable name. These letters indicate which position on the board the variable represents (t=top row, m=middle row, b=bottom row, l=left column, c=center column, r=right column). The values of these variables will always be either “X”, “O”, or “ “ to indicate the space is currently empty. You can change the value of each of these variables to create different board configurations and test the code you will write.

Below where the board is printed to the screen, use if statements to compare the values of the variables to determine and print one of the four possible cases: X has won the game, O has won the game, the game ended in a tie, or the game is not over yet. Consider how you can simplify your code. For example, can you use the and/or/not logical operators and if-elif-else statements to make it shorter and easier to read. We will end up redesigning the game later but the logic that you come up with will still be useful.