Introduction to Programming (ITAS 185)

Lab 3 – Conditional Flow in Python

Date due: Fri, Oct 6, 2023 @ 2:00 pm

## **Learning Objectives**

Upon successful completion of this lab exercise, the student will be able to use python and :

1. Use for and while loops.
2. Use if, elif and else statements.

**To be handed in:**

1. The ***username185L03*** folder should be zipped and uploaded to the ITAS Portal. ***Username*** isyour logon username that is, first name followed by your last name (mine would be allan.mcdonald).

# Checking Marks

To Do:

## Create a python program called grader.py in the folder you created.

## Prompt the user to enter a grade for a student. The value entered is a numeric of type float.

## When the grade is entered, if-elif-else construct shown in class to check the mark and set the result based on the following table:

|  |  |
| --- | --- |
| Mark | Result |
| Less than 60 | Poor |
| Greater than 60 | Passed |
| Equals 100 | Perfect |
| Equals 60 | Squeaked by |
| Less than 0 | Error |
| Greater than 100 | Error |

## Display the mark entered and the result in a sentence such as:

## Your mark of 63 means the result was Squeaked by

## Prompt the user again and continue to prompt the user to enter a new mark until the user enters a mark of 0. When 0 is entered, exit the program.

## Test your program using the numbers in the following table:

|  |  |
| --- | --- |
| Value Entered | Expected Result |
| 59 | Your mark of 59.0 means the result was Poor |
| 60 | Your mark of 60.0 means the result was Squeaked by |
| 61 | Your mark of 61.0 means the result was Passed |
| 100 | Your mark of 100.0 means the result was Perfect |
| 100.003 | Your mark of 100.003 means the result was Error |
| -5.0 | Your mark of -5.0 means the result was Error |

# Guess the Number

To Do:

## Import the random module at the top of the file.

## Using the randint function from the random module, generate a number between the 1 and 100 call that variable guessing\_number.

## Prompt the user to enter a guess and read the value into an integer called my\_guess.

## If the guess is less than the randomly generated number tell the user they are too low. If the guess is greater than the randomly generated number tell the user they are too high.

## Keep prompting the user (in a while loop) to guess the number until the guess equal to the randomly generated number.

## When the guess is correct congratulate the user. For an added challenge (after you complete the rest of the lab), keep track of how many guesses it took the user. An example run is (green is what is entered by the user)…

I am thinking of a number from 1 to 100

What is your guess? 50

50 is too low.

What is your guess? 75

75 is too high.

What is your guess? 62

62 is too high.

What is your guess? 56

Congratulations! That is correct! The number I was thinking of was 56.

# Adding Numbers

To Do:

## In this section you are going to do the same thing twice, once using a for loop and once using a while loop.

## You are going to find the sum of all numbers up to and including the number entered by the user. The algorithm for that is:

1. Prompt user to enter the number.
2. Set the variable the\_sum to 0.
3. Set the loop iterator to 1 (only for a while loop)
4. Loop from 1 to the number entered:
   1. Add the loop iterator value to the\_sum (the\_sum += loop iterator)
   2. Increment the loop iterator (done automatically for for loop)
5. When complete display the number entered and the sum.

## Create a python program called loo.py.

## Prompt the user to enter a positive, non-zero integer value.

## Set the sum to 0.

## Use a for loop with a range to loop from 1 to the value entered and calculate the sum of all the numbers up to that value.

## Display the result to the screen saying that it was done using a for loop.

## Set the sum to 0.

## Set the loop iterator to 1.

## Use a while loop that goes while the loop iterator is less than the number entered

### Add the loop iterator to the sum

### Increment the loop iterator by 1

## Some test data:

|  |  |
| --- | --- |
| Value | Total |
| 4 | 10 |
| 5 | 15 |
| 50 | 1275 |
| 100 | 5050 |

## If you want an extra challenge, repeat from step 4 until the user enters 0.

# Some Basic Error Checking

To Do:

## The distance a vehicle travels can be calculated as follows:

## distance = speed \* time

## For example, if a train travels 40 kilometres per hour (kph) for three hours, the distance traveled is 120 kilometres.

## Create a new python file called distance.py.

## Prompt the user for a speed value (as a float).

## If the speed is less than or equal to 0, display an error message and prompt the user again. Continue this (while) loop until the user enters a value greater than 0.

## Once you have a valid speed, prompt the user for a time value (as a float).

## If the time is less than or equal to 0 OR greater than 24, display an error message and prompt the user again. Continue this (while) loop until the user enters a value in the proper range.

## Once both values are valid, use a loop (I don’t care if you use a for loop or a while loop) to calculate and display distance travelled after EACH HOUR entered.

## An example run would be:

Enter the speed travelled: -10

Invalid speed

Enter the speed travelled: 40

Enter the time travelled: 0

Invalid time

Enter the time travelled: 24.1

Invalid time

Enter the time travelled: 5

Here are your results:

Hours| Distance (km)

0| 0.0

1| 40.0

2| 80.0

3| 120.0

4| 160.0

5| 200.0

NOTICE THE FORMATTING OF THE OUTPUT. Use the f-string formats to display information right-aligned in fields to get the alignment correct. Numbers are to be displayed to 1 decimal place.

## Some test data:

|  |  |  |
| --- | --- | --- |
| Speed | Time | Output |
| 40 | 3 | Hours|Distance  0| 0.00  1| 40.00  2| 80.00  3| 120.00 |
| 53.4 | 5 | Hours|Distance  0| 0.00  1| 53.40  2| 106.80  3| 160.20  4| 213.60  5| 267.00 |
| 4.1234 | 4 | Hours|Distance  0| 0.00  1| 4.12  2| 8.25  3| 12.37 |

# To finish:

## 1. Show your completed lab to me, to be marked.

2. Zip your **entire** folder.

3. Submit the zip file to the ITAS portal.