IC152: Assignment 3
Functions, Loops and Conditions

In this assignment, we will use in-built functions, create user-defined functions, start writing loops, and use conditional statements. You have to write one python file for each question. So, in total there will be 5 python files to submit for this assignment.

If you are solving this assignment in the A11's PC Lab: Keep Fn + F9 pressed during the start of your machine (do not repeatedly press, keep it continuously pressed), and then select the second option with "ubuntu". Please check if you are able to login to moodle, else change the machine.

Problem 1: Using inbuilt functions in python.
a - Import math library and use the following functions on 15.3245 and understand the functionality of each of these functions:

- math.ceil()
- math.floor()
- round()
- type(round())
- format()
- type(format())

The format() function takes two arguments - the floating number itself and the number of decimal places, for e.g '.2f' for 2 decimal places. The first argument to the round() function is a value, and the second argument is the number of decimal places.

Understand the difference between the output of format() and round() using the type() function mentioned above.
Store the output of floor(), ceil() and round() and print the average of the three values.

====Algorithm====

1: Start
2: Import math library
3: Initialize num as 15.3245
4: Apply math.ceil() function
5: Set ceil result = math.ceil(num)
6: Print ”Ceil result: ”, ceil result
7: Apply math.floor() function
8: Set floor result = math.floor(num)
9: Print ”Floor result: ”, floor result
10: Apply round() function
11: Set round result = round(num, 2) ▷ Round to 2 decimal places
12: Print ”Round result: ”, round result
13: Check type of round() result
14: Set round type = type(round result)
15: Print ”Type of round result: ”, round type
16: Apply format() function
17: Set format result = format(num, ’.2f’) ▷ Format to 2 decimal places
18: Print ”Format result: ”, format result
19: Check type of format() result
20: Set format type = type(format result)
21: Print ”Type of format result: ”, format type
22: Compare round() and format() results
23: Print ”Round vs Format: ”, round result, ” vs ”, format result
24: Calculate average of ceil, floor, and round results

25: Set average = (ceil result + floor result + round result) / 3
26: Print ”Average of ceil, floor, and round: ”, average
27: End

b - Use following string methods on different characters and numbers to understand them (e.g., usage is “a”.isalpha()):

- isalpha()
- isnumeric()
- isalnum()
- isspace()

Take a string input from the user and use these functions on the input.
###
Constraints: Strings
Input: Input string
Output: output of all the methods above in separate lines.
Solution :

====**Algorithm**===

1) Take an input from the user of a string and store it in a variable, say **str**.

2) Now you can see above methods use the above methods and store each output in a separate variable (you can use them like **a=str.method()**)

3)Now you can print all the variables in separate lines.

Problem 2 - Create a Python file named `q2Conditions.py` that prompts the user for several inputs and performs different tasks based on those inputs. Implement the following functionalities:

Part a: Prompt the user to input a floating-point number named x. If x is greater than or equal to 10, print `"High"`; otherwise, print `"Low"`.

Answer:

1. **Input:** Prompt the user to enter a floating-point number **x**.
2. **Condition:**
    - If **x** is greater than or equal to 10, print "High".
    - Otherwise, print "Low".
3. **Output:** Print either "High" or "Low" based on the value of **x**.

Algorithm:

1. Start
2. Prompt user for a floating-point number **x**.
3. If **`x >= 10`**, print "High".
4. Else, print "Low".
5. End

Part b: Prompt the user to input an integer named `num`. If `num` is positive, print `"Positive"`; if `num` is zero, print `"Zero"`; if `num` is negative, print `"Negative"`.

Answer:

1. **Input:** Prompt the user to enter an integer **`num`**.
2. **Condition:**
    - If **`num`** is positive, print "Positive".
    - If **`num`** is zero, print "Zero".
    - If **`num`** is negative, print "Negative".
3. **Output:** Print either "Positive", "Zero", or "Negative" based on the value of **`num`**.

    Algorithm:

1. Start
2. Prompt user for an integer **`num`**.
3. If **`num > 0`**, print "Positive".
4. Else if **`num == 0`**, print "Zero".
5. Else if **`num < 0`**, print "Negative".
6. End

Part c: Prompt the user to input a string named `text`. Print whether `text` contains only uppercase letters, only lowercase letters, or a mix of both.

Answer:

Algorithm:

1. **Input:** User enters a string and stores it in a variable named text.
2. **Initialize Variables:** Create two boolean variables, `has_upper` and `has_lower`, and set both to `False`.
3. **Iterate Through Each Character:** For each character `char` in `text`:
    - If `char` is an uppercase letter (use `char.isupper()`):
        - Set `has_upper` to `True`.
    - If `char` is a lowercase letter (use `char.islower()`):
        - Set `has_lower` to `True`.
4. **Result:** After the loop, check the values of `has_upper` and `has_lower`:
    - If both are `True`, print "The text contains a mix of uppercase and lowercase letters."
    - If only `has_upper` is `True`, print "The text contains only uppercase letters."

- If only `has_lower` is `True`, print "The text contains only lowercase letters."
- If neither is `True`, print "The text contains no letters."

Part d: Prompt the user to input the name of a day of the week (e.g., "Monday", "Tuesday", etc.). Print the number of letters in the day name.

Answer:

Algorithm:

1. **Input:** User enters the name of a day of the week and stores it in a variable named `day_name`.
2. **Count Letters:** Use the `len()` function to count the number of letters in `day_name`.
3. **Result:** Print the number of letters in the day name.

Part e: Prompt the user to input three numbers named a, b, and c. Print the average of these three numbers.

Answer:

Algorithm:

1. **Define the function `get_input`:**
   - This function will prompt the user to enter a number.
   - It will return the user input after converting it into a floating-point number.
2. **Define the function `calculate_average`:**
   - This function will take three numbers as input arguments (a, b, and c).

    - It will calculate the average using the formula: average=a+b+c3\text{average} = \frac{a + b + c}{3}average=3a+b+c
    - It will return the computed average.
3. **Define the function `print_output`:**
    - This function will take the average as an input and print it in a formatted way.
4. **In the `main` function:**
    - Call the `get_input` function three times to take the input values for a, b, and c.
    - Call the `calculate_average` function to compute the average using these inputs.
    - Call the `print_output` function to display the result.

Write separate user-defined functions to take and handle input in the function separately for all the parts.

Problem 3: Writing user-defined functions. Write a common python file “q3Functions.py” and :-

a) Write a function that takes the following inputs [6, 2, 20, 496, 30, 8128, 500, 1000, 33550336, 999983] and displays whether they are perfect numbers or not. (A perfect number is a positive integer that is equal to the sum of its proper divisors, excluding itself. For example, 28 is a perfect number because its divisors are 1,2,4,7 and 14 and their sum is 28.)

Ans :)

```
x = [6, 2, 20, 496, 30, 8128, 500, 1000, 33550336, 999983]
# Storing the values of the factors of X
Sum = 0

# Loop till you found all factors
for j in range(0,len(x)-1):
    for i in range(1,x[j]-1):
        if x[j] % i == 0:
            # print(i,"is a factor of ",x)
            Sum += i
    # we are comparing the sum of all factors of the input with the input
    print("\n")
    if Sum == x[j] :
        print(x[j],"is a perfect number")
    else :
        print(x[j],"is not a perfect number")
    Sum = 0
```

Problem 4: Write a user defined function with for **as well as** while loops and conditional statements. Write a common python file “q4Loops.py” and prompt the user with “q4 part a input (integer): ”, for all parts to display the following patterns. The input integers should indicate the dimension of the patterns.

a. Inverted Triangle

   For Input =5

   * * * * *

```
* * * *
  * * *
    * *
      *
```

Solution: Algorithm-

1: **Start**
2: **Prompt** user with ” a input (integer): ”
3: Input integer n (This defines the dimension of the pattern)
4: **Inverted Triangle Using For Loop**:
5: for each row i from 0 to n-1 do
6: Print i spaces
7: Print n-i *(asterisks)
8: end for

9: **Inverted Triangle Using While Loop**:
10: Set i = 0
11: while i ¡ n do
12: Print i spaces
13: Print n-i *(asterisks)
14: Increment i by 1
15: end while
16: **End**

b. Alphabet Triangle
For input =3

A
A B

A B C

You can use any inbuilt function.

Solution-

Algorithm:

1. **Input:**
    - Take an integer `n` as input, representing the number of rows in the triangle.
2. **Initialization:**
    - Set a variable `i` to 1 to represent the current row.
3. **Outer Loop (Row-wise):**
    - While `i` is less than or equal to `n`:
        - Set a variable `j` to 0 to represent the current column.
4. **Inner Loop (Column-wise):**
    - While `j` is less than `i`:
        - Print the character `chr(65 + j)`, which represents the alphabet corresponding to the ASCII code `65 + j`.
        - Increment `j` by 1 to move to the next column.
5. **Newline:**
    - Print a newline to move to the next row.
    - Increment `i` by 1 to move to the next row.

Problem 5: Combining user-defined functions, loops and conditional Statements. Create a Python script named "taskAnalyzer.py". The script should prompt the user with specific input messages for each task like

task A input word, task B input positive number and like these input for for many different parts :

A. Write a function that takes a word input from the user and determines if it is a palindrome. If it is, print “Palindrome,” otherwise, print “Not a Palindrome.” (A word is classified as a palindrome if it reads the same forward and backward. Use your code from the previous assignment if required but remember here you have to define a function).

Answer:

Problem: above
Constraints: Chars and Strings
Input: a string
Output: Input string is palindrome or not
Solution:

- **Algo:**
    Read input string # say we have “abcba”
    Find the middle location
        # 3rd location in “abcba, in general ((length of string) + 1)/2
    Reverse the string on the right of middle location
        # “ba” will become “ab”
    Check if above reversed string is same as string at the left of middle location
        # “ab” of “abcba” is it same as reverse of “ba”

B. Write a function that takes a positive integer N as input from the user and prints whether the number is even or odd. Call the function with different values and corner cases (e.g. if the user gives a negative number or a string or a floating point number, how will you handle it?).

Answer:

Algorithm:

INPUT: Prompt the user to “enter a positive integer”.

Check valid or not:

1. Check if the input can be converted to an integer.
2. If it's not a valid integer (e.g., a string or floating-point number), print an error message (“not an integer”) and ask for a valid input.
3. If the number is negative, print an error message (“integer is not positive”) and ask again for a valid input.

Even or Odd Check:

For valid input , Check whether the input is odd or even. (Can use bit wise operators or modulus).

Output:

Print whether the number is "even" or "odd".

C. Write a script that takes a single word as input from the user and checks through a function call: if the word contains only alphabetic characters (no numbers or special symbols). Print “Alphabet only” if it meets the criteria; otherwise, print “Contains non-alphabetic characters.”

Answer:

Input: A string (a word) provided by the user.

Output:

- **"Alphabet only"** if the word contains only alphabetic characters.
- **"Contains non-alphabetic characters"** if the word contains any numbers or special symbols.

Algorithm:

1. **Take Input:** Read a word from the user.
2. **Check if the word contains only alphabetic characters:**
    - Use Python's string method `.isalpha()` to determine if all characters in the word are alphabets (i.e., no numbers or special characters).
3. **Decision:**
    - If the word contains only alphabetic characters, print "Alphabet only."
    - Otherwise, print "Contains non-alphabetic characters."

D. Write a function that finds and returns the longest word in a given sentence. (Call the function with the user's input and print the longest word.)

Answer:

Problem: above
Constraints: Chars and Strings
Input: a string (sentence)
Output: Longest word in the given sentence.

Solution:

- **Algo:**

Read input string # say we have “I like python”

Initialize

- Longest_word = “”
- Max_length = 0
- count = 0
- word = “”

Iterate in the entire string:

If current character is a space:

If count > Max_length:

Update Max_length to count

Update Longest_word to word

Reset count = 0

Reset word = “”

else (current character is not a space):

Concatenate the current character to word

Increase the count by 1

After the iteration is over, check the condition for last word:

If count > Max_length:

Update Max_length to count

Update Longest_word to word

Return the Longest_word

E. Write a function that counts and returns the number of words in a given phrase. (Take the user's input and call the function to print the total number of words.)

Problem: above
Constraints: Chars and Strings
Input: a string
Output: a number
Solution:

Algorithm:

#Initialize:
- word_count = 0 (variable to store count of words)
- in_word = false (to indicate if we're currently in a word)

#Iterate:
- For each character char in phrase:
    If char is a space and in_word is true:(meaning we reached the end of the word)
        word_count++(increment)
        in_word = false
    Otherwise:(we are inside the word)
        in_word = true

#Final check:
- If in_word is true:(meaning the last character was part of a word)
    word_count++(increment)

#Return:
- word_count

Create the folder having your python files, with name having your roll number followed by "_assignment3" (don't use inverted commas in folder name), compress the folder with .zip extension and submit it on moodle.

Make sure that you delete all your files from the lab PC/Laptop, and shut it down before you leave.