Programming in Python 2023/24

Assignment 01 *APRIL 2024*

Topic: from introduction to Control Structures

INSTRUCTIONS

This Assignment covers **From introduction Control Structures Chapter.** Each question should be answered in a different file. You should submit the python file and a **pdf** file where you have captured the outputs (ONLY OUTPUTS) after running the questions solutions and other files you have used in this assignment including those provided with the assignment. All this should be in a zipped folder with your name and reg (example: myname_12345).

Do not use packages pandas, or user defined functions in answering the questions.

Submission deadline: Latest by `Monday 6th May 2024 at 1900 hrs. Submit through your CR.`

Question 01

1. A palindrome is a number or a text phrase that reads the same backwards or forwards. For example, 12321, RACECAR are palindromes. Write a program that reads in a **positive** five-digits **integer** and determines whether it is a palindrome. The program should restrict the user to enter positive five-digits integer. If not, the user should be given an error message: `Entered integer …. is not a five digit integer, try again.` If number is not positive, the user should be given an error message: `Entered integer …. is not positive, try again` and then re-prompted to re-enter the integer. Sample display:

```
Enter five-digits integer: ………..
Entered integer is pandrome / not palindrome
```

Test your program with following integers
(a) 345 (b) -65456 (c) 65456 (d) 45678 (e)12321

Question 02

(a) Write a program using **for loop** to calculate and display in 6 decimal places the sum of the first N terms of the following series where N is a positive integer (N>0) specified by user. You can also include `while loop` in restricting N value to be greater than zero but **for**

loop must appear in your code. When the program is run, user will be prompted to enter value of N.

$$Sn=\frac{1}{2}+\frac{2}{3}+\frac{3}{4}+\frac{4}{5}+....$$

Test your program with: (i) N=0 (ii) N=1 (iii) N=15 (iv) N=100 (v) 1000

(b) Repeat (a) using while loop(s)

(c) Using nested loops of your choice, write a program to generate the following pattern:

```
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Question 03

Write a program that reads an unspecified number of positive and negative integers (one number at a time). Reading of inputs ends when the input is `999` and `999` should not be counted (it is used as `sentinel` to signal the end of the input). Zero (0) should be treated as positive.

The program should determine and display how many positive and negative values have been read, average of positive numbers, average of negative numbers, maximum of positive numbers, and maximum of negative numbers. Display the average to two decimal places. If there is no any positive number or negative number then display the message "`No any Positive number has been entered`" or "`No any negative number has been entered`" when you are displaying info.

If the user enters 999 **only**, the program should display the message “`You didn't enter any number, program terminated`” and then the program terminates.

Sample display:

```
Enter an integer, end inputs by 999: 1
Enter an integer, end inputs by 999: -23
Enter an integer, end inputs by 999: 47
Enter an integer, end inputs by 999: 94
Enter an integer, end inputs by 999: -34
Enter an integer, end inputs by 999: 999
```

```
==== Positive numbers info ====
The number of positives is:    3
The sum of positives is:       142
The average of positives is:   47.33
Maximum value of positives is: 94

==== Negative numbers info ====
The number of negatives is:     2
The sum of negatives is:        -57
The average of negatives is:    -28.50
Maximum value of negatives is:  -23
```

Test your program with following data:

```
(a)  15, 22, 49, -9, 57, -45
(b)  23, 199, 89, 12, 0
(c)  -23, -199, -89, -12
(d)  -12, 98, -42, -38
(e)  999
```

Question 04

Write a **menu** driven program to enable user to issue commands. When the program is run it displays the following menu:

```
Following services are available:

1. Start the motor
2. Stop the motor
3. Display help
4. Exit

   Enter your choice (1..4): …..
```

Choice no 1 displays the message: `Motor started`
Choice no 2 displays the message: `Motor stopped`
Choice no 3 displays the following three lines:

```
Press 1 to start the motor
Press 2 to stop the motor
Press 4 to exit the program
```

If choice is outside the range, then error message `Wrong choice, please re-enter your choice` should be displayed and use re-enters another choice.
The program continues to run until the user enters 4 when the program terminates and displays the message `Program Terminated`

`Extra: The question` can be improved. If user presses 2 and the motor is not yet started, the message should be displayed that the motor is not yet started.

Question 05

The file `stdsscores_file.py` (provided) contains a list (with the name `stdsscores`) of student scores with each row in the provided list contains a data for each student in the format:

```
[SN, Test1, Test2, UE]
```

The scores (out of 100) are for Undergraduate students at EASTC in 2 tests and UE for a certain course. Two tests (Test1 and Test2) contribute to Continuous Assessment (CA). Weights for Test1 and Test2 in CA are 45% and 55% respectively. Final Mark (FM) out of 100 is calculated as: FM=0.4*CA+0.6*UE where both CA and UE are out of 100.

Grades ranges are:

```
A: 70≤ FM ≤ 100;  B+: 60 ≤ FM < 70;  B: 50 ≤ FM < 60
C: 40 ≤ FM < 50; D: 35 ≤ FM < 40; E:0 ≤ FM <35
```

Pass mark is grade **C**.

Remark is either P or F where P stands for Pass and F for Fail.

Write a program to perform the following services. You need to **import** the list in your workspace. Answers should be obtained programmatically **NOT** by manual means:

(a) Display how many students in the list.

(b) Create a copy of `stdsscores` list and call it `fscores.` Display the last 5 records from `fscores.`

Use `fscores` to answer following questions:

`(c)` Calculate `CA`, `FM` (in 1 decimal place), `Grade` and `Remark` for each student and update `fscores`. That means each row of `fscores` will have following format in specified order: `[SN, Test1, Test2, UE, CA, FM, Grade, Remark]` and display the first five records.

(d) Display minimum, maximum, and average for Test1, Test2, CA, UE and FM. Averages should be in 2 decimal places. Display the values in neat tabular form as follows:

```
             Test1  Test2  CA    UE    FM
Minimum        x      x     x     x     x
Maximum        x      x     x     x     x
Average        x      x     x     x     x
```

(e) Display record(s) of student(s) who scored lowest mark and also student(s) who scored the highest mark in the FM. Note that there is a possibility of having more than one in each category.

(f) Create a dictionary called `gradesSummary` which contains number of students for each grade. Sample of `gradesSummary` is `(in sorted order of grades)`

```
gradesSummary={'A':...,'B':...., 'B+':...,'C':...,'D':....,
'E':...}.
```

(g) Display grade(s) with the highest number of students and grades with lowest number of students. State also number of students in those grades. Sample display:

```
Grade(s) with the lowest number of students are:……. and number
of students is: ……….. 
Grade(s) with the highest number of students are:……. and number
of students is: ……….. 
```

(h) Display record of a student whose S/N is specified from the Keyboard. If the serial number specified is not in the allowable range, display the following error message and user should be re-prompted to enter S/N.

```
Wrong entry! Valid ranges are … up to ….. try again.
```

(i) It has been found that one student was forgotten in the list and has to be added at the end of `fscores.` His scores are: `Test1=45.6, Test2=76.4, UE=65.5.` Add that record to `fscores` and display the last records from `fscores` and check if the record corresponds to added student.

(j) Display the best top 5 students based on FM in the neat tabular form as follows (where x stands for values.

```
SN      FM
x       x
x       x
…..     ……
x       x
```

=== *END* ===