ISYS 812
Programming and Applications for Data Analytics

Practice Session 4

Question 1

We will write a program that will convert temperatures between Fahrenheit and Celsius, depending on the user input. The program will have a menu asking the user the type of conversion wanted, then ask for the input temperature, perform and display the conversion.

Reminder :

Conversion from Fahrenheit to Celsius: **C** = 5/9 x (**F** - 32)

Conversion from Celsius to Fahrenheit: **F** = **C** x 9/5 + 32

The program must contain the following 3 functions:

display_menu: This function receives no parameter and returns the user choice.

It displays a menu allowing the user to pick the temperature conversion they want to perform: **1.**Fahrenheit to Celsius or **2.**Celsius to Fahrenheit. The user choice (1 or 2) should be returned by the function.

convert_temp: This function receives one parameter representing the user choice.

This function asks the user for the temperature input and perform the appropriate conversion depending on the user choice made in the **display_menu** function. The converted temperature is displayed.

main : This function receives no parameter and returns no value.

This function is the starting point of the program and should perform the following operations:

1. Call **display_menu** function to display the program menu and obtain the user conversion choice

2. Call **convert_temp** function to perform the conversion and display the converted temperature

The following are examples of the program output. User inputs are in **bold**.

```
MENU
=========
Indicate which temperature conversion you would like to perform, by entering
1 or 2
    1. Fahrenheit to Celsius
    2. Celsius to Fahrenheit

Enter your choice, 1 or 2: 1
Enter degrees Fahrenheit 99
The converted temperature is 37.22 degree Celsius
```

```
MENU
=========
Indicate which temperature conversion you would like to perform, by entering
1 or 2
    1. Fahrenheit to Celsius
    2. Celsius to Fahrenheit

Enter your choice, 1 or 2: 2
Enter degrees Fahrenheit 38
The converted temperature is 100.4 degree Celsius
```

Question 2

Modify Question 1 so that you the program ask the user if another temperature requires to be converted. If the user enters “yes” or “y”, the program will re-run until no more temperature need to be converted.

Question 3

In this exercise we will write a complete script to derive the final grade of students enrolled in an Intro to Python course. The course grading is composed of 3 Homework, 2 Midterms and the Final Exam.

Each homework has a maximum grade of 20. Midterms and final exam have a maximum grade of 100.

The assignments coefficient is as follows: Homework: 25%,   Midterm : 50%   Final Exam : 25%

The grading scale is as follows: A: 100-90   B: 89-80   C: 79-70   D: 69-60   F: 59-0

For this script, **each student is represented by a dictionary** with three keys being the different assignment type: **Homework**, **Midterm** and **Final Exam.** Homework and Midterm grades are represented by a **list** of values. The Final Exam is a single value.

Our script must composed of the following functions:

enter_grade : This function receives 3 parameters :

- **assignment** is a string representing the type of assignment (Homework, Midterm or Final Exam)
- **amount** is an integer representing the quantity of grades for a given assignment
- **max_grade** is an integer representing the maximum possible grade for a given assignment. Not that Midterm and Final Exams have a maximum grade of 100 while Homework have a maximum grade of 20. Your function should set **max_grade default value as 100**

This function asks the user to enter a set of grades for a given assignment types. You should make sure that the specific **amount** of grade is asked to the user and that each grade is between 0 and **max_grade.**

The grades are saved in **a list that is returned by the function.**

Remember that list***.append()*** allows you to add a value at the end of a list.

setup_grade: This function receives a **dictionary** as parameter. This dictionary represents a single student. For each assignment type, the function is calling the **enter_grade** function to get the user to enter the grades. This function does not return any value.

display_grade : This function receives a **dictionary** as parameter and display all the grades for a given student.

average_grade: This function receives a **list** of grades and **return the average grade** of this list.

course_grade : This function receives **2 lists and a single value** as parameters. The first list corresponds the student homework grades, the second list corresponds to the student midterm grades and the last parameter is the student final exam grade. ***You may need to make sure you are passing a single value.***

The function should first call the **average_grade** function to retrieve the average Homework grade and the average Midterm grade.

The function should then **compute and return the final course grade** using each assignment coefficients (Homework is 25%, Midterm is 50% and Final Exam is 25%) . Note that Homework maximum grades is 20.

letter_grade : This function receives an value **CourseGrade** as parameter representing the total grade for a given student. Based on the grading scale (A: 100-90   B: 89-80   C: 79-70   D: 69-60   F: 59-0) the function will display the corresponding student letter grade.

This function does not return any value.

For each of those function as a documentation right below the function header. (You can just copy paste the function description)

You can test your functions using the data of the following students
(Any resemblance to actual real persons is purely coincidental)

	Homework			Midterms		Final Exam
	HW1	HW2	HW3	Exam 1	Exam 2	
Leyla	15	13	18	75	77	88
Sameer	15	14	12	68.5	40	55

Hint : Since the grades must be entered by the user, you can create dictionaries with empty lists and value 0 for the final exam.

The following is an example of the program output. User input are in **bold**.

Leyla

```
You will enter the Homework grades. Make sure they are between 0 and 20
Enter the grade for Homework 1 15
Enter the grade for Homework 2 13
Enter the grade for Homework 3 18
You will now enter the Midterm grades. Make sure they are between 0 and 100
Enter the grade for Midterm 1 75
Enter the grade for Midterm 2 105
Error: The Midterm grade should be between 0 and 100
Enter the grade for Midterm 2 77
You will now enter the Final Exam grade. Make sure it is between 0 and 100
Enter the grade for the Final Exam 88
For this student, the Homework grades are :  [15.0, 13.0, 18.0]
For this student, the Midterm grades are : [75.0, 77.0]
For this student, the Final Exam grade is : 88.0
With a total of  79.16666666666667 the student grade is  C
```

Sameer

```
You will enter the Homework grades. Make sure they are between 0 and 20
Enter the grade for Homework 1 15
Enter the grade for Homework 2 14
Enter the grade for Homework 3 12
You will now enter the Midterm grades. Make sure they are between 0 and 100
Enter the grade for Midterm 1 68.5
Enter the grade for Midterm 2 40
You will now enter the Final Exam grade. Make sure it is between 0 and 100
Enter the grade for the Final Exam 55
For this student, the Homework grades are :  [15.0, 14.0, 12.0]
For this student, the Midterm grades are : [68.5, 40.0]
For this student, the Final Exam grade is : 55.0
With a total of  57.958333333333333 the student grade is  F
```

Question 4 : Map and Filter

In this exercise, we will discuss two built-in functions than can be used in data-analysis : **map()** and **filter().**

1. Let's start by creating a function called **times3.** This function takes a number as parameter, multiply this value by 3 and return the result.

2. Now set a list **seq1to10** that contains all the integers between 1 and 10.

3. Using a **loop** and calling the function **times3,** create a new list **newseq** that contains all the values of **seq1to10** multiplied by 3.

Your new list must contain [3, 6, 9, 12, 15, 18, 21, 24, 27, 30]

map ()

4. The **map() function** gives you're an alternative to the previous step.

map() official description (Shif+Tab) is **:**

```
map(func, *iterables) --> map object

Make an iterator that computes the function using arguments from

each of the iterables. Stops when the shortest iterable is exhausted.
```

In other (or better) words, **map()** takes two parameters : a function and a sequence of item (for example, a list). For each item of the sequence**, map()** call the function and record the result**.**

Based on those explanation, let's use the map function to replace our previous loop and see the output:

```
map (times3, seq1to10)
```

The previous function does not display the results but instead indicate **the address in memory** where the results are.

To display the results, we need to **cast** the result of the map() function as a list :

```
list (map(times3, seq1to10) )
```

Which should now display our new list

5. Let's go a step further by defining the **lambda expression** of the function **times3.**

We saw in class that lambda expressions are useful to write short functions. They can also be used within the map() function.

6. In your previous expression `list (map(times3, seq1to10) )` replace **times3** by the lambda expression. You should get the same result.

filter ()

7. Create a function **evenNum** that take a value as parameter and returns **True** if the value is even.

filter() official description (Shif+Tab) is **:**

```
filter(function or None, iterable) --> filter object

Return an iterator yielding those items of iterable for which function(item)
is true. If function is None, return the items that are true.
```

The filter() function takes as parameters: a sequence and a **condition** in the form of function. The filter() function then return all the elements from the sequence that satisfy the condition (the elements from which the function return True)

Based on this description, lets display all the even value from **`seq1to10:`**

`filter(evenNum, seq1to10)` display the address of the resulting list

`list(filter(evenNum, seq1to10))` display the desired output : `[2,4,6,8,10]`

8. Just like we did for the map function, we can directly write the function lambda expression within the filter function to obtain the same input.