Questions for Practice

1. Question

A person has five friends namely Andrew, Barker, Charlee, David and Eric. Write a solution in Python to create a list ***friends*** using these names and define a function ***check_friend*** that prompt the user to enter a name to check if he/she is a friend of him.

Sample Input Output: 1

```
>>>Enter a name: John
John is not your friend
```

Sample Input Output: 2

```
>>>Enter a name: Charlee
Charlee is your friend
```

2. Question

A shopkeeper would like to find the daily average sales for all the items sold by her and record them in a file. She needs a program to do so. Write a Python program with following functions in order to meet her requirement:

i. A function called `getDailySales` that reads in sales amount for some items sold in a grocery shop for a particular day. The function should store the sales values in a *list* called `sales` and return it.

ii. A function called `totalDailySales` that accepts `sales` *list* and calculate the total sales for a particular day. The function should store the total sales values in a variable called `total` and return it.

iii. A function called `avgDailySales` that accepts `sales` *list* and calculate the average sales for a particular day. The `avgDailySales` should call the `totalDailySales` function for calculating the total sales before finding the average. The function should store the average sales value in a variable called `average` and return it.

iv. A function called `avgWeeklySales` that calls the `getDailySales` function for 7 days. The `avgDailySales` function should also be called after calling the `avgDailySales` function to find the average sales of each day. The `avgWeeklySales` function should then write the average sales for each day in the "Average_Sales.txt" file.

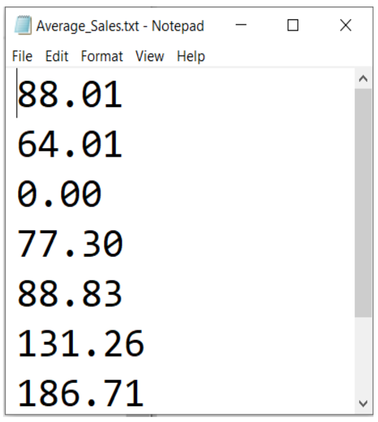
After writing all these functions, call the `avgWeeklySales` function in your Python program. Sample input of the program is as follows:

```
Finding average sales for day 1

Enter sales amount for the first item [-1 to end]: 89.00

Enter sales amount for the next item [-1 to end]: 58.50

Enter sales amount for the next item [-1 to end]: 14.10

Enter sales amount for the next item [-1 to end]: 199.65

Enter sales amount for the next item [-1 to end]: 78.80

Enter sales amount for the next item [-1 to end]: -1
```

```
Finding average sales for day 2

Enter sales amount for the first item [-1 to end]: 144.20

Enter sales amount for the next item [-1 to end]: 23.25

Enter sales amount for the next item [-1 to end]: 12.70

Enter sales amount for the next item [-1 to end]: 75.90

Enter sales amount for the next item [-1 to end]: -1


Finding average sales for day 3

Enter sales amount for the first item [-1 to end]: -1


Finding average sales for day 4

Enter sales amount for the first item [-1 to end]: 45.50

Enter sales amount for the next item [-1 to end]: 125.35

Enter sales amount for the next item [-1 to end]: 89.70

Enter sales amount for the next item [-1 to end]: 48.65

Enter sales amount for the next item [-1 to end]: -1

Finding average sales for day 5

Enter sales amount for the first item [-1 to end]: 120.00

Enter sales amount for the next item [-1 to end]: 98.45

Enter sales amount for the next item [-1 to end]: 58.35

Enter sales amount for the next item [-1 to end]: 89.90

Enter sales amount for the next item [-1 to end]: 88.55

Enter sales amount for the next item [-1 to end]: 77.70

Enter sales amount for the next item [-1 to end]: -1
```

```
Finding average sales for day 6

Enter sales amount for the first item [-1 to end]: 78.50

Enter sales amount for the next item [-1 to end]: 125.00

Enter sales amount for the next item [-1 to end]: 360.10

Enter sales amount for the next item [-1 to end]: 47.10

Enter sales amount for the next item [-1 to end]: 98.35

Enter sales amount for the next item [-1 to end]: 78.50

Enter sales amount for the next item [-1 to end]: -1


Finding average sales for day 7

Enter sales amount for the first item [-1 to end]: 564.00

Enter sales amount for the next item [-1 to end]: 52.00

Enter sales amount for the next item [-1 to end]: 21.35

Enter sales amount for the next item [-1 to end]: 211.20

Enter sales amount for the next item [-1 to end]: 85.00

Enter sales amount for the next item [-1 to end]: -1
```

Take note that there are no sales recorded on day 3 in the example above as the shop was closed due to Public Holiday. Text file written for the above inputs is given in Figure 1 below:



```
88.01
64.01
0.00
77.30
88.83
131.26
186.71
```

Figure 1: “Average_Sales.txt” File

3. Question

A short distance runner is preparing for the upcoming Olympic game. He went through multiple trial sessions with his coach. The coach recorded the distance ran by the runner and the time taken for the same during each trial session. After all trial sessions, the coach needs to find the runner's speed during each trial session and record all these details in a text file. Write a Python program with the following functions to fulfill the requirement of the coach:

i. A function called `getDistanceTime` that reads in the distance ran in meters and time taken in seconds for a particular trial session. The function should store the distance and time of multiple trial sessions in a single *list* called `distanceTime` and return it.

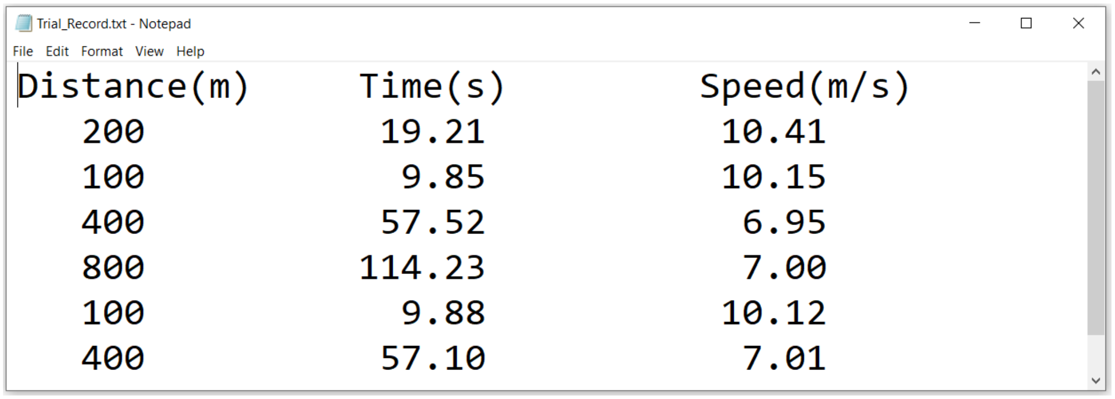
ii. A function called `calculateSpeed` that accepts the `distanceTime` *list* and calculate the speed for each pair of distance and time for all the trial sessions. The function should store the speed values in another *list* called `speed` and return it.

iii. A function called `trialRecord` that calls the `getDistanceTime` and `calculateSpeed` functions. The `trialRecord` function should then record all details in "Trial_Record.txt" file. Each line in the text file should contain distance ran, time taken and speed for each training session.

After writing all these functions, call the `trialRecord` function in your Python program. Sample input of the program is as follows:

```
Enter distance ran in meters(m) [-1 to end]: 200

Enter time taken in seconds(s): 19.21

Enter distance ran in meters(m) [-1 to end]: 100

Enter time taken in seconds(s): 9.85

Enter distance ran in meters(m) [-1 to end]: 400

Enter time taken in seconds(s): 57.52

Enter distance ran in meters(m) [-1 to end]: 800

Enter time taken in seconds(s): 114.23

Enter distance ran in meters(m) [-1 to end]: 100

Enter time taken in seconds(s): 9.88
Enter distance ran in meters(m) [-1 to end]: 400

Enter time taken in seconds(s): 57.10

Enter distance ran in meters(m) [-1 to end]: -1
```

Text file written for the above inputs and outputs is given in Figure 1 below:



```
Distance(m)      Time(s)          Speed(m/s)
   200           19.21              10.41
   100            9.85              10.15
   400           57.52               6.95
   800          114.23               7.00
   100            9.88              10.12
   400           57.10               7.01
```

Figure 1: “Trial_Record.txt” File

4. Question

An automobile servicing workshop provides car maintenance and repair services to its customers. Everyday, the manager of the workshop records the car number, labour charge and cost of parts after servicing each car. The manager needs a program to bill his customers. The bill is subject to 6% of government tax. Finally, he would to record all these details in a text file. Write a Python program with following functions in order to meet his requirement:

i. A function called `getDailyService` that reads in car number, labour charge and cost of parts after servicing each car for a particular day. The function should store all these values in a *list* called `service` and return it.

ii. A function called `calculateDailyBill` that accepts `service` *list* and calculate total bill for each car for a particular day. Every bill should include 6% of government tax. The function should store the bill amount of each car serviced for a particular day in a *list* called `bill` and return it.

iii. A function called `weeklyService` that calls the `getDailyService` function for 7 days. The `calculateDailyBill` function should also be called after calling the `getDailyService` function to calculate the bill for every car service in each day. The `weeklyService` function should then write every service and bill details in the "Weekly_Service.txt" file.

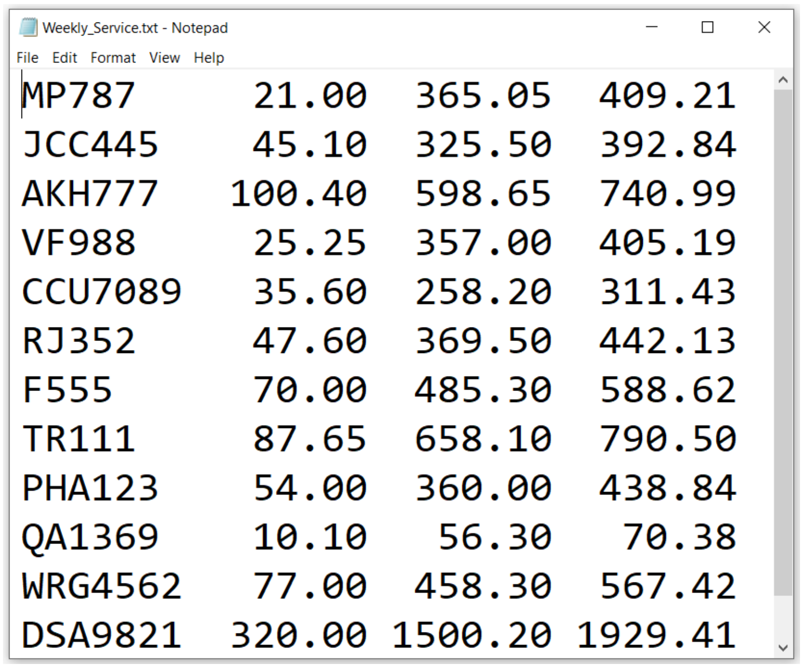
After writing all these functions, call the `weeklyService` function in your Python program. Sample input of the program is as follows:

```
Services done in day 1

Enter first car's number [-1 to end]: MP787

Enter labour charge: 21.00

Enter cost of parts replaced: 365.05

Enter next car's number [-1 to end]: JCC445

Enter labour charge: 45.10

Enter cost of parts replaced: 325.50

Enter next car's number [-1 to end]: -1
```

```
Services done in day 2

Enter first car's number [-1 to end]: AKH777

Enter labour charge: 100.40

Enter cost of parts replaced: 598.65

Enter next car's number [-1 to end]: -1


Services done in day 3

Enter first car's number [-1 to end]: VF988

Enter labour charge: 25.25

Enter cost of parts replaced: 357.00

Enter next car's number [-1 to end]: CCU7089

Enter labour charge: 35.60

Enter cost of parts replaced: 258.20

Enter next car's number [-1 to end]: RJ352

Enter labour charge: 47.60

Enter cost of parts replaced: 369.50

Enter next car's number [-1 to end]: -1


Services done in day 4

Enter first car's number [-1 to end]: F555

Enter labour charge: 70.00

Enter cost of parts replaced: 485.30

Enter next car's number [-1 to end]: -1
```

```
Services done in day 5

Enter first car's number [-1 to end]: TR111

Enter labour charge: 87.65

Enter cost of parts replaced: 658.10

Enter next car's number [-1 to end]: -1


Services done in day 6

Enter first car's number [-1 to end]: PHA123

Enter labour charge: 54.00

Enter cost of parts replaced: 360.00

Enter next car's number [-1 to end]: QA1369

Enter labour charge: 10.10

Enter cost of parts replaced: 56.30

Enter next car's number [-1 to end]: WRG4562

Enter labour charge: 77.00

Enter cost of parts replaced: 458.30

Enter next car's number [-1 to end]: -1


Services done in day 7

Enter first car's number [-1 to end]: DSA9821

Enter labour charge: 320.00

Enter cost of parts replaced: 1500.20

Enter next car's number [-1 to end]: -1
```

Text file written for the above inputs is given in Figure 1 below:



```
MP787      21.00  365.05  409.21
JCC445     45.10  325.50  392.84
AKH777    100.40  598.65  740.99
VF988      25.25  357.00  405.19
CCU7089    35.60  258.20  311.43
RJ352      47.60  369.50  442.13
F555       70.00  485.30  588.62
TR111      87.65  658.10  790.50
PHA123     54.00  360.00  438.84
QA1369     10.10   56.30   70.38
WRG4562    77.00  458.30  567.42
DSA9821   320.00 1500.20 1929.41
```

Figure 1: “Weekly_Service.txt” File

5. Question

A college has appointed you as a programmer to computerize their assessment activities. As a first assignment, you are required to write a program in Python to allow instructors to create quiz for the module that they are teaching. The questions and right answer for each question should be permanently written in a file. On the other hand, the program should allow students to take the quiz any number of times. Following guide can help you to build the solution:

i. Write a function called `createQuiz` that reads in quiz questions, options outlined under each of them and the correct answer for each of them as well. The function should store all these values in a *list* called `quiz` and return it.

ii. Write a function called `saveQuiz` that accepts `quiz` *list* and write it into a file called *quiz.txt*.

iii. Write a function called `takeQuiz` that reads the *quiz.txt* file. The function then should display each question on the screen and prompt for an input from the user. Ultimately the function should display the total score of the user on the screen.

iv. Write a function called `menu` to provide options for users to select. Ultimately your program should call `menu` function and other functions are called in the `menu` function based on user's selection.

Note: You may create quiz on any module/subject of your interest. There should be at least 5 questions created for the quiz with four options (option A, B, C and D) under each of them.

Take Extra exercise: [optional]

1. Expand your program so that it can work for more than one module
2. Create login options so only Admin/Lecturers are allowed to create quizzes and students can only take quizzes.
3. Record best score for each student in a file.

6. Question

An AC servicing workshop provides AC maintenance and repair services to its customers. Everyday, the manager of the workshop records the AC number, labour charge and cost of parts after servicing each AC. The manager needs a program to bill his customers. The bill is subject to 5% of government tax. Finally, he would to record all these details in a text file. Write a Python program with following functions in order to meet his requirement:

i. A function called `getDailyService` that reads in AC number, labour charge and cost of parts after servicing each AC for a particular day. The function should store all these values in a *list* called `service` and return it.

ii. A function called `calculateDailyBill` that accepts `service` *list* and calculate total bill for each AC for a particular day. Every bill should include 5% of government tax. The function should store the bill amount of each AC serviced for a particular day in a *list* called `bill` and return it.

iii. A function called `weeklyService` that calls the `getDailyService` function for 7 days. The `calculateDailyBill` function should also be called after calling the `getDailyService` function to calculate the bill for every AC service in each day. The `weeklyService` function should then write every service and bill details in the "Weekly_Service.txt" file.

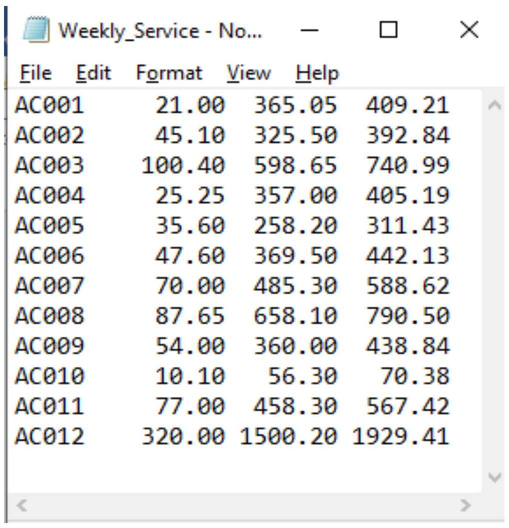
After writing all these functions, call the `weeklyService` function in your Python program. Sample input of the program is as follows:

```
Services done in day 1

Enter first AC's number [-1 to end]: AC001

Enter labour charge: 21.00

Enter cost of parts replaced: 365.05

Enter next AC's number [-1 to end]: AC002

Enter labour charge: 45.10

Enter cost of parts replaced: 325.50

Enter next AC's number [-1 to end]: -1
```

```
Services done in day 2

Enter first AC's number [-1 to end]: AC003

Enter labour charge: 100.40

Enter cost of parts replaced: 598.65

Enter next AC's number [-1 to end]: -1


Services done in day 3

Enter first AC's number [-1 to end]: AC004

Enter labour charge: 25.25

Enter cost of parts replaced: 357.00

Enter next AC's number [-1 to end]: AC005

Enter labour charge: 35.60

Enter cost of parts replaced: 258.20

Enter next AC's number [-1 to end]: AC006

Enter labour charge: 47.60

Enter cost of parts replaced: 369.50

Enter next AC's number [-1 to end]: -1


Services done in day 4

Enter first AC's number [-1 to end]: AC007

Enter labour charge: 70.00

Enter cost of parts replaced: 485.30

Enter next AC's number [-1 to end]: -1
```

```
Services done in day 5

Enter first AC's number [-1 to end]: AC008

Enter labour charge: 87.65

Enter cost of parts replaced: 658.10

Enter next AC's number [-1 to end]: -1


Services done in day 6

Enter first AC's number [-1 to end]: AC009

Enter labour charge: 54.00

Enter cost of parts replaced: 360.00

Enter next AC's number [-1 to end]: AC010

Enter labour charge: 10.10

Enter cost of parts replaced: 56.30

Enter next AC's number [-1 to end]: AC011

Enter labour charge: 77.00

Enter cost of parts replaced: 458.30

Enter next AC's number [-1 to end]: -1


Services done in day 7

Enter first AC's number [-1 to end]: AC012

Enter labour charge: 320.00

Enter cost of parts replaced: 1500.20

Enter next AC's number [-1 to end]: -1
```

Text file written for the above inputs is given in Figure 1 below:



```
AC001      21.00  365.05  409.21
AC002      45.10  325.50  392.84
AC003     100.40  598.65  740.99
AC004      25.25  357.00  405.19
AC005      35.60  258.20  311.43
AC006      47.60  369.50  442.13
AC007      70.00  485.30  588.62
AC008      87.65  658.10  790.50
AC009      54.00  360.00  438.84
AC010      10.10   56.30   70.38
AC011      77.00  458.30  567.42
AC012     320.00 1500.20 1929.41
```

“Weekly_Service.txt” File