Assignment 2

- Rules same as Assignment 1, e.g. to be done individually (discussing the problems with each other is fine), code to be uploaded, marks, bonus marks policy, etc.
- Only some problems are given now - so you can get started. More problems will be added next week.
- Any changes/clarifications to the problems will be highlighted in blue.
- For these problems, as before you will have to create your test data.
- If any details are missing - you can make assumptions about them and state them as comments at the top of the program.

==

Question 1.
The canteen in the Institute maintains has a table of prices of items, like:

1. Samosa: 15
2. Idli: 30
3. Maggie: 50
4. Dosa, 70

   …

For the program you have to write, set the menu in your program by this statement (feel free to add more items).

menu = [("Samosa", 15), ("Idli", 30), ("Maggie", 50), ("Dosa", 70), ("Tea", 10), ("Coffee", 20), ("Sandwich", 35), ("ColdDrink", 25)]

Maggie, 1, Rs 50
Samosa, 5, Rs 75

TOTAL, 6 items, Rs 125

Question 2.
Write a program that will repeatedly take (for a student) input of this type: course_no, number_of_credits, and grade_received (eg. CSE101 4 A). These inputs are for the courses the student has done in the semester. If no input is given (i.e. just a return), that means no courses are left. From this data, print the transcript for the semester for the student as:

Course_no: grade
Course_no: grade

….

SGPA: n.nn (two decimal places)

In the transcript, the records should be printed such that they are sorted by the course_no (hint: when sorting a list of tuples/lists, the default sorting is with the first element of the tuple/list).

The course numbers are an alphanumeric string with capital letters in the start and digits at the end (e.g. CSE101, CSSS21), credits can be 1, 2 or 4, and grade can be A+ (10), A(10), A-(9), B(8), B-(7), C(6), C-(5), D(4), F(2) (the number in () is their numeric equivalent for SGPA calculation.) The SGPA is computed as (sum of: credits*grade/ total_credits). If any input is not valid, print a suitable message ("improper course no", "incorrect credit", or "incorrect grade") and ignore that input.

Suggestions: A list of tuples/lists may be a good structure to use. Read the input as tuples into a list. When inputs are done, process the list.
First just write code to read the input, split it into a list, and pass it to a function to check for errors (recall string has operations like isdigit() to check if the string is integer). If input is valid, add to a list of tuples (initially, to check the course name structure just check if it is an alphanumeric string - later write a function to check its valid structure - requires a bit of logic). Independently write a function which, given a list of tuples, can compute the SGPA. Then combine the two.

Question 3:
Before you graduate, you get the signature from all your friends in your yearbook - which will be stored as a dictionary for the program. In this dictionary, the keys are the name of other students in the batch; the value is either 0 (for not signed) or 1 (for signed). Like you, all other students are also doing the same - and there is a dictionary entry for each. For creating this dictionary, input is given in a file - if there are N students, then the file contains:

name1:
name2, 1
name3, 0
name4, 1
…
name2:
name1, 0
name3, 0
name4, 0
…

(Like this data is there for all the graduating students).

Write a program to determine who has the most signatures and who has the least (if there are more than one for max/min - print all).

Suggestion. Initially manually create a dictionary yearbook = {name1: {...}, name2: {....}, …}, and then determine students with most/least signatures. Then write a function to read the file and create this dictionary.

Question 4.
The goal of this game is for the user (player) to guess the 5 character word in a limited number of chances.

Pick up at least 50 5 character words (e.g. from https://7esl.com/5-letter-words/) and save them in a list or any other structure (you can assign them in a statement, i.e. hard code them). Select a word at random from this list - for this, generate a random number between 0 and length of the list of words using the random number generator randint() provided in python (Pls look it up - it is quite easy) - this is the index in the list for the word the user has to guess.

Now prompt the user to input a five character string as his/her guess. After the guess, the program should show the user the chars from the guess string which are in correct places, and correct chars in wrong places, by outputting:

--a-d (if a and d were in the input string and are in these two places in the word),
other characters present: b (if b is present in the word)

Let the user repeatedly give guesses, till either the guess is correct, or the number of tries is 6.

Bonus: Accept from the user only valid words as guesses. For this, use an available online dictionary API to check if the given string is a word or not. And if not, prompt the user and do not count this attempt.

Question 5.
You are given a list of coordinates (each as a x,y tuple) representing a 2D shape. If the shape has N coordinate, create a Nx3 matrix with the first column having the x values, 2nd column having the y values, and the third column being 1 for all. This represents the matrix for the shape.

You have to scale this 2D shape. For scaling, take as input cx and cy (scaling parameters) from the user, and form a matrix of the type:

$$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For scaling the shape, multiply the matrix of the shape with this matrix. Finally, return the new shape in the form of a list of coordinates - which will be the first two columns of the resultant matrix.

Suggestion. While you can easily find matrix multiplication on the net, try to first write your own function for this - use online help only if needed.

Question 6 (program for grading).
For IP course, the performance in different elements of the class for students is given in a file (IPmarks.txt) - one line per student as comma separated values:

Roll_no, m1, m2, m3, …

Roll_no is a string, while m1, m2, … are integer marks the student got in assessment 1, 2, …

Separately, the weight of each of the assessments is given as a list of tuples (you can hard code this in your program - assume that correct number of items is provided):

wts = [(10, 5), (20, 5), (100, 15), (40, 10), …]

Where the first item is the maximum marks of the assessment, and the second is the weight in % this assessment carries in the final evaluation (the sum of weights should be 100). In the above, assessment 1 (say, a quiz) has maximum marks of 10 and its weight in the final total is 5%; assessment 3 has max marks of 100, and its weight is 15).

For each student, compute the weighted sum of marks normalized to 100. (For weighted sum, assessment 1 marks can be normalized to 5, assessment 3 to 15, …), and the grade, and write it in a file (IPgrades.txt) as:

Roll_no, total_marks, grade

For grading, assume that A is above 80%, A- from 70, B from 60, B- from 50, C from 40, C- from 35, D from 30, and below 30 is F.

Question 7 (Your personal address book):
Write a program to create your own address book, in which each entry has: Name, address, phone number, and email address. In the program, the address book should be a dictionary - the key of the dictionary has to be the **name** of the person, while the value has to be another dictionary with **address**, **phone**, and **email** as its keys. Initially, you can assume that people in your directory have unique names. Then modify the program to handle multiple persons with same name.(One way to handle this is to keep name as the key, but the value may be a list of dictionaries - one for each person)

Your program should provide the following operations: (i) insert a new entry, (ii) delete an entry (iii) find all matching entries given a partial name, (iv) find the entry with a phone number or email, and (v) exit. When the program exits, it writes the current dictionary in a file (addrbook.txt) and when it is started again next time, it reads the existing address book.

Suggestion: To create an address book with some entries, write a small script which will call the add function to add a few entries to create a dictionary.

Bonus. Write a program to merge address book of your friend with yours. For this, you can store the address book as a json or as a list of dictionary items - if both you and your friend agree to the structure, merging will be easier.

Question 8. (A simplified page rank system.)
Given a text file (pages.txt) which has lines of the form:

URLnn, init_importance: text containing some URLnn.

Each line represents a page with the first URLnn being its URL and init_importance is a number between 0 and 1.0, which gives the initial importance of this page. The URLnn in the rest of the line refers to the URL links to other pages that this page contains. (To simplify, instead of giving a full path, we are using URLnn, and instead of having separate files for each page, we are giving the text of a page as a line in the file). Example:

URL00, 0.5: This is page zero, and has references to URL01, URL09, and also to URL08. It may have repeated references - so there are two references to URL09.
URL02, 0.6: This is another page (page is represented as a line in this). This has reference to URL05, URL04, and URL00
…

Pages are ranked according to their overall importance. Let the total number of unique links in a page i be links[i] (i.e. these many unique pages this page refers to). The overall importance of a page i is sum over all the pages (j) which have a link to page i of: init_importance[j]/ links[j] (i.e. all the pages to which the page j has a link are distributed the importance of this page equally)

Your program has to find the highest ranking N pages (N can be set in a variable or taken as input) for a given input file.

It seems natural to create a dictionary with page URL as the index, and having its init_importance, overall importance, the set of URLs it accesses, etc in it.

Note. This simplified version has only one round of computation for the overall importance from the initi_importance. The ranking algorithm actually takes the current importance (starting with init_importance, which is often set to 1) and then updates the importance repeatedly till the changes are very small. If you want, you can extend your program to implement this.

Bonus problem/mini-project.

You can work in groups of 2 (or 3 max), i.e. all students can submit the same code - at the top of the code have a comment stating names of all students who worked together.
Write an interesting application using some public APIs. The application should be interactive - i.e. gives the user some query options, and gives the result of the query after fetching the data using APIs and processing it. There should be more than one API call in this application, preferably from different base APIs (i.e. APIs from different organizations).

A suggested mini/project - if you have a good working application for this, let the Instructor know by email (he wants to use it).
Using the music sites' APIs, develop an application that can get the list of songs that were sung by some <singer> whose lyrics were written by some <poet/lyrics_writer>. Either of the fields, if not specified, should give songs for all singers/poets. Give this list as a CSV specifying the song (generally by its name or the starting lines - whatever the site uses), the singer, the lyrics_writers/poet, the year it was sung, and whatever other info is there. (It will be good if songs from multiple sites are in this list - will make it more complete.) (It will be nice if you can also give a YouTube link for listening/viewing this song).