CP1404/CP5632 Assignment 1 – Travel Tracker 1.0

Task:

You are to write a Python (3) program, as described in the following information and sample output. This assignment will help you build skills using selection, repetition, file input/output, exceptions, lists, functions and string formatting. Do not define any of your own classes or use constructs that haven't been taught in this subject. Assignment 2 will build on this with more advanced constructs including dictionaries, classes and a Graphical User Interface (GUI).

Some requirements have in-text help references, like [0], that refer to the resources list near the bottom of this document. Please check these references to find help on that topic. ***Everything*** you need to complete this assignment can be found in the subject materials.

Start your work by clicking this link to create a new repository in GitHub Classroom:

https://classroom.github.com/a/aw7JLb1S

Do not use any other repo or a copy of this one... just use this actual repository!
This will give you a new repo containing starter files including a README for your project, all of which you must use. Do not add any other files in this project, and do not rename anything - just use this as your assignment repo. Do not "download" this repo, but rather *checkout* this repo using PyCharm ("Get from Version Control").

Program Overview:

This program is a simple "travel tracker" that allows a user to track places they wish to visit and places they have already visited. The program reads and writes a list of places in a text file, and each place has:

- name, country, priority, whether it is visited (v) or unvisited (n)

Users can choose to see the list of places, including the total number of places and unvisited places. The list will be sorted by visited status then by priority (decreasing number). [1]
Did you see that reference? Look below for [1] and you'll find out where we have taught you how to sort lists by multiple keys.
Users can add new places and mark places as visited.
They cannot change places from visited to unvisited.

Program Functionality Details:

Ensure that your program has the following features, as demonstrated in the sample output below. Your program should:

- display a welcome message with your name in it
- display a menu for the user to choose from [2, 3]
- error-check user inputs as demonstrated in the sample output [4]
- load a CSV (Comma Separated Values) file of places (just once at the very start); a sample CSV file is provided for you and you must use this format [5] (note: you're not expected to use the csv module, but you're welcome to)
- *when the user chooses **list**:* display a neatly formatted (lined up) list of all the places with their details and the number of places left to visit [4]. Unvisited places should have a * next to them. Note that the lining up is dynamic, based on the longest place name and country [6]
- *when the user chooses **recommend**:* display a random choice from any available unvisited places
  - if there are no unvisited places, then "No places left to visit!" should be displayed
- *when the user chooses **add**:* prompt for the place's name, country and priority, error-checking each of these [3], then add the place to the list in memory (not to the file); new places are always unvisited

- *when the user chooses **mark visited**:* display the list of all places (same as for the list option), then allow the user to choose one place (error-checked), then change that place to visited
    - if there are no unvisited places, then "No unvisited places" should be displayed
- *when the user chooses **quit**:* save the places to the CSV file, overwriting the file contents (note that this should be the only time that the file is saved)

Coding Requirements and Suggestions:

- Work incrementally on this task: complete small parts of it at a time rather than trying to get it all working at once.
- You are assessed on your use of version control including commits and commit messages, so please commit regularly (each logical chunk or milestone) and use meaningful commit messages in the imperative voice, as taught in class. Your commits should show steady work completed over reasonable time, not all in a short period.
- Edit the module docstring at the very top of your code file to contain your own details.
- Make use of named constants as appropriate (e.g., for the characters that represent the song's learned/unlearned status).
- Use functions appropriately for each significant part of the program: this is the divide-and-conquer problem-solving approach. Follow the principles you've learned about functions, including the single responsibility principle (SRP).
- ***Only load (read) the places file once***, when the program starts.
- ***Only save (write) the places file once***, when the program ends.
- ***Store the place data in a list of lists*** and pass that to any functions that need access to it. Note: this variable should not be global. The ***only*** global variables you may have are CONSTANTS. (Did you understand this? If you use global variables, your functions will be poorly designed. **Do not use any global variables.**)
- Do not store a place's index – this is just its position in the list.
- The menu choice should handle uppercase and lowercase letters.
- Use exception handling where appropriate to deal with input errors (including entering numbers and selecting places). You should be able to use generic, customisable functions to perform input with error checking (e.g., getting the place name and country can reuse the same function).
- The output shows that the solution does not require correct plurals (e.g., "1 places"). You are welcome to leave yours this way. You may add logic to print these statements correctly, but it is not expected or assessed.

Check the rubric carefully to understand how you will be assessed. There should be no surprises here – this is about following the best practices we have taught in class.

Output Requirements:

Sample output from the program is provided below. **Ensure that your program matches this, including spaces, spelling, and the formatting of the place lists.** Think of this as helpful guidance as well as training you to pay attention to detail. The sample output is intended to show a large (but maybe not exhaustive) range of situations including user input error handling.

Integrity:

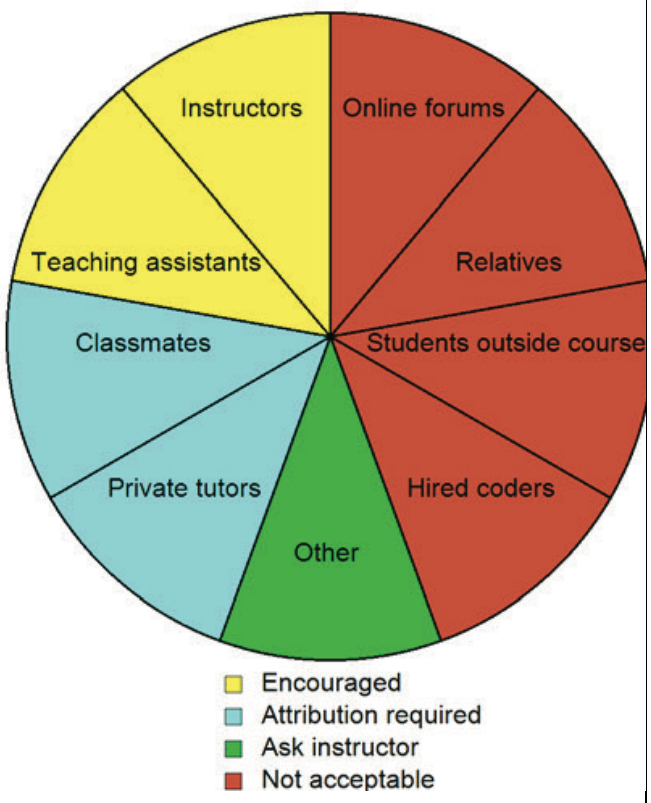
The work you submit for this assignment must be your own. Submissions that are detected to be too similar to that of another student or other work (e.g., code found online) will be dealt with according to the College procedures for handling plagiarism and may result in serious penalties.

The goals of this assignment include helping you gain understanding of fundamental programming concepts and skills, and future subjects will build on this learning. Therefore, it is important that you develop these skills to a high level by completing the work and gaining

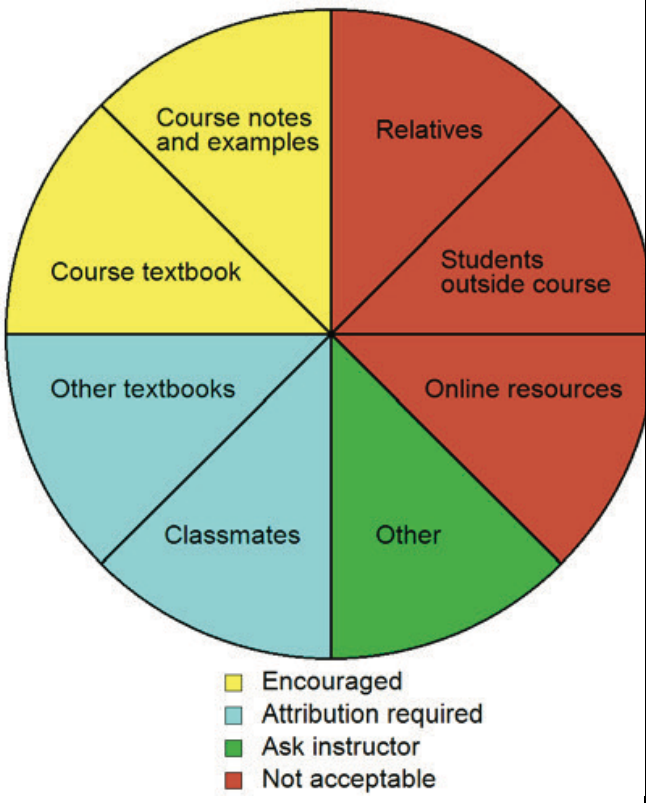
the understanding yourself. You may discuss the assignment with other students and get assistance from your peers, but you may not do any part of anyone else's work for them and you may not get anyone else to do any part of your work. Note that this means you should **never give a copy of your work to anyone or accept a copy of anyone else's work, including looking at another student's work or having a classmate look at your work**. If you require assistance with the assignment, please ask **general** questions on the discussion forum, or get **specific** assistance with your own work by talking with your lecturer or tutor.

The subject materials (lecture notes, practicals, textbook and other guides provided in the subject) contain all of the information you need for this particular assignment. You should not use online resources (e.g., Stack Overflow or other forums) to find resources or assistance as this would limit your learning and would mean that you would not achieve the goals of the assignment - mastering fundamental programming concepts and skills.

Assistance: Who can you get help from? Use this diagram to determine from whom you may seek help with your programs. [7]



Resources: Where can you get code from? Use this diagram to determine where you may find code to use in your programs.



Submission:

Zip up your project/repo directory as it is, then submit this zip file. Ensure that you do NOT include a venv. Name the file like: **FirstnameLastnameA1.zip**. Upload your single zip file on LearnJCU under Assessment (click on the title of the assignment).

Due:

Submit your assignment by the date and time specified on LearnJCU. Submissions received after this date will incur late penalties as described in the subject outline.

Sample Output:

The following sample run was made using a CSV file that contained:

```
Lima,Peru,3,n
Auckland,New Zealand,1,v
Rome,Italy,12,n
```

You should be able to figure out what parts of the sample output below are user input.

```
Travel Tracker 1.0 - by Lindsay Ward
3 places loaded from places.csv
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> no
Invalid menu choice
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> l
*1. Lima     in Peru          3
*2. Rome     in Italy        12
 3. Auckland in New Zealand   1
3 places. You still want to visit 2 places.
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> R
Not sure where to visit next?
How about... Lima in Peru?
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> m
*1. Lima     in Peru          3
*2. Rome     in Italy        12
 3. Auckland in New Zealand   1
3 places. You still want to visit 2 places.
Enter the number of a place to mark as visited
>>> 0
Number must be > 0
>>> 4
Invalid place number
>>> no
Invalid input; enter a valid number
>>> 3
You have already visited Auckland
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> m
*1. Lima     in Peru          3
*2. Rome     in Italy        12
 3. Auckland in New Zealand   1
3 places. You still want to visit 2 places.
Enter the number of a place to mark as visited
>>> 1
Lima in Peru visited!
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> L
*1. Rome     in Italy        12
 2. Auckland in New Zealand   1
 3. Lima     in Peru          3
3 places. You still want to visit 1 places.
Menu:
L - List places
```

```
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> m
*1. Rome     in Italy        12
 2. Auckland in New Zealand   1
 3. Lima     in Peru          3
3 places. You still want to visit 1 places.
Enter the number of a place to mark as visited
>>> 1
Rome in Italy visited!
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> l
 1. Auckland in New Zealand   1
 2. Lima     in Peru          3
 3. Rome     in Italy        12
3 places. No places left to visit. Why not add a new place?
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> m
No unvisited places
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> r
No places left to visit!
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> a
Name:
Input can not be blank
Name:
Input can not be blank
Name: Malaga
Country:
Input can not be blank
Country: Spain
Priority: 2
Malaga in Spain (priority 2) added to Travel Tracker
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> l
*1. Malaga   in Spain         2
 2. Auckland in New Zealand   1
 3. Lima     in Peru          3
 4. Rome     in Italy        12
4 places. You still want to visit 1 places.
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> a
Name: Xanadu
Country: Atlantis
Priority: 1
Xanadu in Atlantis (priority 1) added to Travel Tracker
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
```

```
Q - Quit
>>> r
Not sure where to visit next?
How about... Xanadu in Atlantis?
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> L
*1. Xanadu   in Atlantis      1
*2. Malaga   in Spain         2
 3. Auckland in New Zealand   1
 4. Lima     in Peru          3
 5. Rome     in Italy        12
5 places. You still want to visit 2 places.
Menu:
L - List places
R - Recommend random place
A - Add new place
M - Mark a place as visited
Q - Quit
>>> q
5 places saved to places.csv
Have a nice day :)
```

At the end of this run, the saved CSV file contained:

```
Xanadu,Atlantis,1,n
Malaga,Spain,2,n
Auckland,New Zealand,1,v
Lima,Peru,3,v
Rome,Italy,12,v
```

References – Resources from Subject Materials:

Selected subject materials are referenced here to help you find guidance for specific parts of the assignment (e.g., sorting a list by multiple values is covered in [1] and you will find a template for writing menus for console programs in [2]).
General references are not listed specifically but should be obvious (e.g., file input/output is covered in the lecture and practical on files).
You should find the programming patterns "cheat sheet" helpful for a number of things:
https://github.com/CP1404/Starter/wiki/Programming-Patterns

1. itemgetter from Chapter 7 - Lists and Tuples.
2. Practical 01 - PyCharm, Control. https://github.com/CP1404/Practicals/tree/master/prac_01
3. Programming Patterns. https://github.com/CP1404/Starter/wiki/Programming-Patterns
4. Practical 02 - Strings, Files, Exceptions. https://github.com/CP1404/Practicals/tree/master/prac_02
5. Chapter 5 - Files and Exceptions 1.
6. Practical 05 - Dictionaries, Code Reviews with PRs. https://github.com/CP1404/Practicals/tree/master/prac_05
7. Negotiating the Maze of Academic Integrity in Computing Education. https://dl.acm.org/citation.cfm?doid=3024906.3024910

Marking Scheme:

Ensure that you follow the processes and guidelines taught in class in order to produce high quality work. Do not just focus on getting the program working. This assessment rubric provides you with the characteristics of exemplary, good, competent, marginal and unacceptable work in relation to task criteria.

<table>
<tr><th>Criteria</th><th>Exemplary (9, 10)</th><th>Good (7, 8)</th><th>Satisfactory (5, 6)</th><th>Limited (2, 3, 4)</th><th>Very Limited (0)</th></tr>
<tr><td>Correctness
Worth double</td><td>Program works correctly for all functionality required.</td><td rowspan="9">Exhibits aspects of exemplary (left) and satisfactory (right)</td><td>Program mostly works correctly for most functionality, but there is/are some required aspects missing or that have problems.</td><td rowspan="9">Exhibits aspects of satisfactory (left) and very limited (right)</td><td>Program works incorrectly for all functionality required.</td></tr>
<tr><td>Error checking</td><td>Invalid inputs are handled well using exceptions and control logic as instructed, for all user inputs.</td><td>Invalid inputs are mostly handled correctly as instructed, but there is/are some problem(s), e.g., exceptions not well used.</td><td>Error checking is not done or is very poorly attempted.</td></tr>
<tr><td>Similarity to sample output (including all formatting)</td><td>All outputs match sample output perfectly, or only one minor difference, e.g., wording, spacing.</td><td>Multiple differences (e.g., typos, spacing, formatting) in program output compared to sample output.</td><td>No reasonable attempt made to match sample output. Very many differences.</td></tr>
<tr><td>Identifier naming</td><td>All function, variable and constant names are appropriate, meaningful and consistent.</td><td>Multiple function, variable or constant names are not appropriate, meaningful or consistent.</td><td>Many function, variable or constant names are not appropriate, meaningful or consistent.</td></tr>
<tr><td>Use of code constructs</td><td>Appropriate and efficient code use, including good logical choices for data structures and loops, good use of constants, etc.</td><td>Mostly appropriate code use but with definite problems, e.g., unnecessary code, poor choice of data structures or loops, no use of constants.</td><td>Many significant problems with code use.</td></tr>
<tr><td>Use of functions</td><td>Functions and parameters are appropriately used, functions are well reused to avoid code duplication.</td><td>Functions used but not well, e.g., incorrect/missing parameters or calls, unnecessary duplication or main code outside main function.</td><td>No functions used or functions used very poorly.</td></tr>
<tr><td>Formatting</td><td>All formatting is appropriate, including correct indentation, horizontal spacing and consistent vertical line spacing. PyCharm shows no formatting warnings.</td><td>Multiple problems with formatting reduces readability of code. PyCharm shows formatting warnings.</td><td>Readability is poor due to formatting problems. PyCharm shows many formatting warnings.</td></tr>
<tr><td>Commenting</td><td>Helpful block/inline comments and meaningful docstrings for all functions, top docstring contains all program details (name, date, basic description, GitHub URL).</td><td>Comments contain some noise (too many/unhelpful comments) or some missing program details in top docstring or some inappropriate or missing block/inline comments.</td><td>Commenting is very poor either through having too many comments (noise) or too few comments.</td></tr>
<tr><td>Use of version control</td><td>Git/GitHub used effectively and the repository contains a good number of commits with good messages that demonstrate incremental code development.</td><td>Aspects of the use of version control are poor, e.g., not many commits, meaningless messages that don’t represent valuable incremental development.</td><td>Git/GitHub not used at all.</td></tr>
</table>