Assignment 2 (A2)

Due date: Friday, March 3, 2023 (11:59pm)
Submission via Git only

Overview

Programming environment

For this assignment you must ensure your code executes correctly on the virtual machine or the programming environment (for M1\M2 computers) you configured as part of Lab 01. This is our "reference platform". This same environment will also be used by the teaching team when evaluating your submitted work. You will have to make sure that your program executes perfectly on the reference platform. If your programs do not run on reference platform, your submission will receive 0 marks.

All test files and sample code for this assignment are available in the 'a2' folder of your git repository and you must use the git pull command to download a copy of the files:

```
git pull
```

Individual work

This assignment is to be completed by each individual student (i.e., no group work). You are encouraged to discuss aspects of the problem with your fellow students. However, sharing of code fragments is strictly forbidden. Note SENG 265 uses highly effective plagiarism detection tools to discover copied code in your submitted work. Both, using code from others and providing code to others, are considered cheating. If cheating is detected, we'll abide by the strict UVic policies on academic integrity: https://www.uvic.ca/library/help/citation/plagiarism/

Learning objectives

- Learn or review basic features of the Python 3 programming language.
- Use the Python 3 programming language to write an introductory data science project.
- Use Git to manage changes in your source code and annotate the evolution of your solution with messages provided during commits. Remember, the only acceptable way of submitting your work is using Git.
- Test your code against the provided test cases.

route_manager.py: Descriptive analytics about airline routes data

a) In A2, route_manager is a small program that uses relevant Python structures and libraries to process airline routes data to produce descriptive analytics.

b) Based on provided arguments and data files (i.e., datasets), route_manager.py will help us answer the following questions:

> **q1:** What are the top 15 airlines that offer the greatest number of routes with destination country as Canada?
>
> **q2:** What are the top 30 countries with least appearances as destination country on the routes data?
>
> **q3:** What are the top 10 destination airports?
>
> **q4:** What are the top 10 destination cities?
>
> **q5:** What are the unique top 10 Canadian routes (i.e., if CYYJ-CYVR is included, CYVR-CYYJ should not) with most difference between the destination altitude and the origin altitude?

c) The format of the arguments used for the program is similar to the one used for Assignment 1 (A1). A formal description of the inputs is described at the end of this document.

d) The output of your program **WILL NOT** be directed to stdout. After each execution, **your program must generate two files**:

   a. **q<X>.csv:** A file compliant with the CSV File format that contains data in a table-based format to answer the question passed as an argument to the program (e.g., **q1**, **q2**). Please

note that the '<x>' placeholder in the name of the file corresponds to the question passed as an argument. For example, **q1.csv, q2.csv, q3.csv, q4.csv**, and **q5.csv.**

b. **q<X>.pdf:** This is a file that contains an image with a graph\figure (e.g., bar plot or pie plot, depending on the argument passed) that allows to visualize the data in **q<X>.csv** and answer the question passed as an argument to the program (e.g., **q1**, **q2**). Please note that the '<x>' placeholder in the name of the file corresponds to the question passed as an argument. For example, **q1.pdf**, **q2.pdf**, **q3.pdf**, **q4.pdf**, and **q5.pdf.**

e) The most reliable way (and the only one encouraged) to test your program is to use the provided **tester** file which will validate the output produced by your program, given a particular question.

Restrictions

- Your program must run as a script.
- Your program must be decomposed into easy-to-understand components. Good program decomposition is required. Methods or functions must be short and effectively parameterized.
- Unwieldy functions are not accepted. The main function should especially be easy to understand and represent a decomposition of the problem at hand.
- Typing (i.e., type hints) must be used in variables and functions defined in your program.
- Do not use global variables.
- **You can only use Python modules and libraries that DO NOT require additional installations on the reference platform**. As part of the installation and configuration of the reference platform, we included relevant libraries for managing data such as numpy and pandas. The latter might be very useful to answer the assignment questions (i.e., **q1**, **q2**, **q3**, **q4**, and **q5**). For visualization purposes, matplotlib was also installed. For your reference, you can find all the Python modules\libraries installed in lines 54-64 in the Vagrantfile. **Failing to comply with this restriction will result in significant lost marks or even 0 marks for the assignment.**
- Keep all your code in one file (**route_manager.py**) for this assignment. In Assignment 4 we will use the multi-module features of Python.

Testing your solution

- This time the **tester** file will execute your program automatically. You'll only need to pass the number of the question as an argument (e.g., `./tester 1`). If no arguments are passed to the tester file (i.e., `./tester`), it will run the tests for all the questions. Using a specialized library that compares the differences between .csv files, the **tester** file will describe the differences between the expected output and the one provided by your program.
- Refer to the example commands in the file "TESTS.md" for appropriate command line input. Your solution must accommodate all specified command line inputs.
- Make sure to use the test files provided as part of this assignment.
- Use the test files and listed test cases to guide your implementation effort. Develop your program incrementally. Save stages of your incremental development in your Git repository.

- For this assignment you can assume all test inputs are well-formed (i.e., exception handling is not required).
- **DO NOT rely on visual inspection**. You can use the provided **tester** file so you can verify the validity of your outputs in a simpler manner.

What to submit

A single Python source file named "route_manager.py" submitted (i.e., Git push to your remote repository) to the a2 folder in your repository.

Grading

Assignment 2 grading scheme is as follows. In general, straying from the assignment requirements might result in zero marks due to automated grading.

Scheme

- **(30%) # Tests Passed**
- **(30%) # Appropriate Graphs\Figures Generated**
- **(40%) Code Qualitative Assessment**
    - **(20%) Functional decomposition, program-scope or file-scope variables:** quality coding requires the good use of functions. Code that relies on few large functions to accomplish its goals is considered poor-quality code. Typically, a good program has a main function that does some basic tasks and calls other functions that do most of the work.
    - **(20%) Code structure:** The code style must be consistent, uniform, and facilitate readability throughout the file.
    - **(15%) Proper naming conventions:** You must use proper names for functions and variables. Using random or single character variables is considered improper coding and significantly reduces code readability. Single character variables as loop variables is fine.
    - **(15%) Typing:** to facilitate readability and maintainability in your solution, typing (i.e., type hints) is required for this assignment when declaring variables and functions.
    - **(10%) Debugging/Comment artifacts:** You must submit a clean file with no residual commented lines of code or unintended text.
    - **(10%) Documentation and commenting:** the purpose of documentation and commenting is to write information so that anyone other than yourself (with knowledge of coding) can review your program and quickly understand how it works. In terms of marking, documentation is not a large mark, but it will be part of the quality assessment.
    - **(10%) Quality of solution:** marker will access the submission for logical and functional quality of the solution. Some examples that would result in a reduction of marks: solutions that read the input files several times, solutions that represent the data in inappropriate data structures, solutions which scale unreasonably with the size of the input.

Scale

A grade		B grade		C grade		D grade		F grade		
grade	marks	grade	marks	grade	marks	grade	marks	grade	marks	description
A+	90-100	B+	77-79							Either no submission given, or submission represents little work or none of the tests pass. No submission, 0 marks. Submissions that do not run, 0 marks. Submissions that fail all tests and show a poor to no effort (as assessed by the marker) are given 0 marks. Submissions that fail all tests, but represent a sincere effort (as assessed by the marker) may be given a few marks
A	85-89	B	73-76	C+	65-69					
A-	80-84	B-	70-72	C	60-64	D	50-59	F	0-49	

Input specification

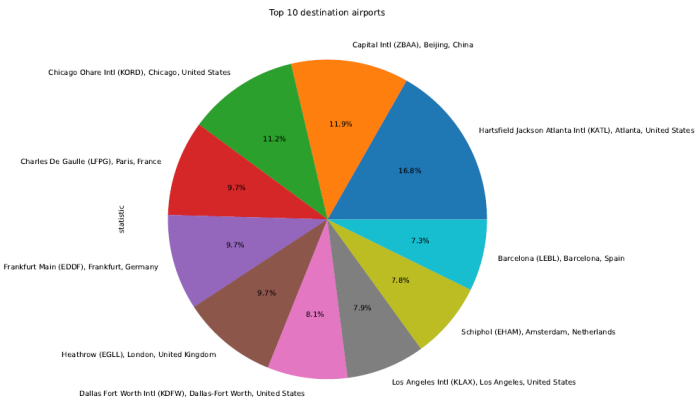
- All input files are in the .YAML format (i.e., airlines.yaml, airports.yaml, routes.yaml) and come from the "Worlds Airports and Airlines Datasets" dataset in Kaggle.com.
- YAML is a simple data-representation format that is widely used across the software engineering and data science lifecycles. We only need basic concepts from the YAML format for this assignment. You can refer to online resources (e.g., this 5-min video) for a brief overview of the language.
- Please refer to the A1's write up or the original data source for a description of the fields in the input data files.

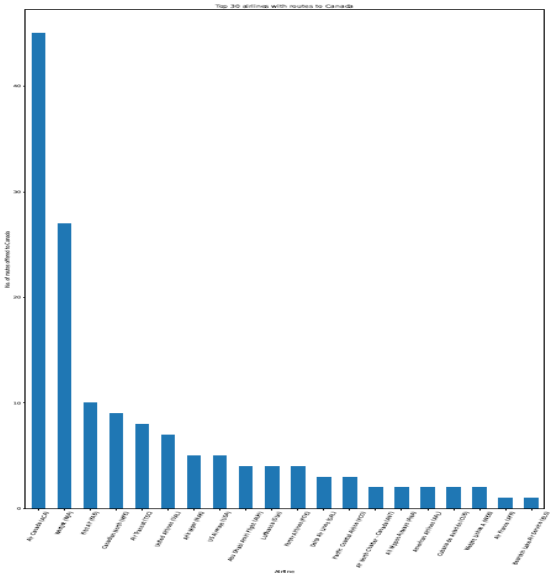
Program Arguments

Argument	Description	Example
--AIRLINES	Location of the airlines.yaml file.	`./route_manager.py --AIRLINES="airlines.yaml" --AIRPORTS="airports.yaml" --ROUTES="routes.yaml" --QUESTION="q1" --GRAPH_TYPE="bar"`
--AIRPORTS	Location of the airports.yaml file.	
--ROUTES	Location of the routes.yaml file.	
--QUESTION	Question (e.g., **q1**, **q2**) to be answered by the program.	
--GRAPH_TYPE	Either "bar" or "pie" indicates the type of graph to be generated	

Output specification

- After execution, your program must produce\create the following files:
  - **A. q<X>.csv**: This file will contain two-dimensional data (i.e., a table with two columns) that represents the answer to the question passed as an argument to the program (e.g., **q1**, **q2**, **q3**, **q4**, and **q5**).
    - The first column in the table (i.e., X axis) must be named 'subject'.
    - The second column in the table (i.e., Y axis) must be named 'statistic'.
    - Examples of the expected output.csv files for each test are provided (i.e., tests\test01.csv, tests\test02.csv, etc.).
  - B. **q<X>.pdf:** This is a file that contains an image with a graph\figure (e.g., bar plot or pie plot) that allows to visualize the data in **q<X>.csv** and answer the question passed as an argument to the program (e.g., **q1**, **q2**, etc.).
    - Note that, to improve computing performance, we reduced the size of the data for the routes file by removing random samples. Thus, results obtained during A2 differ from reality (e.g., top destination airports)[1].
    - Only one image is necessary for each test.
    - The effectiveness of the image generated to answer the question asked to the program will be evaluated. Thus, choose and design your plots carefully. Include plot title, labels, axis names, and other relevant information in the plots to make sure that the question can be answered using your visualization. Be creative.
    - Examples of plots generated using the matplotlib module are presented below.





[1] If you are interested on running your program using the complete dataset, you can download the data using the following command: `scp NETLINKID@seng265.seng.uvic.ca:/seng265work/2023-spring/a2/* .`