CITS1401 Computational Thinking with Python
Project 1 Semester 2 2021

Project 1: (updated 1st September 2021)

Submission deadlines: 5:00 pm, Friday 17th September 2021
Value: **15%** of the total marks of CITS1401.

To be done individually.

You should construct a Python 3 program containing your solution to the following problem and submit your program electronically on Moodle. The name of the file containing your code should be your student ID e.g. `12345678.py`. No other method of submission is allowed. Your program will be automatically run on Moodle for sample test cases provided in the project sheet if you click the "check" link. However, your submission will be tested thoroughly for grading purposes after the due date. Remember you need to submit the program as a single file and copy-paste the same program in the provided text box. You have only one attempt to submit so don't submit if you are not satisfied with your attempt. All open submissions at the time of the deadline will be automatically submitted. There is no way in the system to open the closed submission and reverse your submission.

You are expected to have read and understood the University's guidelines on academic conduct. Following this policy, you may discuss with other students the general principles required to understand this project, but the work you submit must be the result of your own effort. Plagiarism detection, and other systems for detecting potential malpractice, will therefore be used. Besides, if what you submit is not your own work then you will have learned little and will, therefore, likely, fail the final exam.

You must submit your project before the submission deadline listed above. Following UWA policy, a late penalty of 5% will be deducted for each day (24 hours), after the deadline, that the assignment is submitted. No submissions will be allowed after 7 days following the deadline except approved special consideration cases.

Overview

Social media is one of the most attractive marketing platform nowadays. We are spending more time on social media than ever which is estimated to be 15% of our waking lives. Australians are spending one-third of the their online time on social media and over 1 in 3 users turn to social media to gather information about brands they are interested in. More interesting facts can be found here.

Therefore, the advertisements need to be more relevant and interesting to the audience. Many different factors are considered to adapt the advertisements for each and every user.

In this project, you are required to write a computer program that can read the data from a CSV (comma-separated values) file provided to you and return different interesting analytical results. Your program should follow the following specification.

Specification: What your program will need to do

INPUT

Your program must define the function `main` with the following signature:

```
def main(inputFile,queryLocId,d1,d2):
```

The input arguments are:

- `inputFile` is the name of the CSV file containing the information and record about the location points which need to be analysed for this project. The first row of the CSV file contains the following headers:
    - LocId: The ID of a location point.
    - Latitude: The latitude of location point.
    - Longitude: the longitude of location point.
    - Category: Location Types which can be of only one of the five types: Parking (P), Hospital (H), Restaurant (R), Chemist Shop (C) and Super Market (S).
- `queryLocId` is a location id for which we are analysing the record. This input argument will accept a string.
- `d1` and `d2` are the input arguments that provide the dimension of rectangular boundary around the input argument `queryLocId`. `d1` will extend the rectangular region in the East-West direction, whereas, `d2` will extend the rectangular region in North-South with respect to `queryLocId`. The rectangular region created by the parameters `d1` and `d2` will be considered as the search space. For example, if the latitude and longitude of a location point L10 are given as (x,y), and the value of input parameters are `d1` and `d2`. In this case, the search space will create a rectangle region where the coordinate of the North-East (NE), North-West (NW), South-West (SW), and South-East (SE) corners will be NE = (x+`d1`, y+`d2`), NW = (x-`d1`, y+`d2`), SW = (x-`d1`, y-`d2`), SE = (x+`d1`, y-`d2`) respectively. Both of these input arguments (`d1`,`d2`) will be numeric data.

OUTPUT

The function is required to return the following outputs in the order provided below:

- List containing the locations (which can be found under the heading LocId in CSV file) which fall inside the rectangular region formed by the parameters `d1` and `d2` with respect to the input location id `queryLocId`.
- List containing the locations (which can be found under the heading LocId in CSV file) of the same location category as the `queryLocId` found inside the above mentioned rectangular region.
- list containing the distances of the locations (considering the locations points as cartesian coordinates https://en.wikipedia.org/wiki/Cartesian_coordinate_system), in ascending order from the `queryLocId` which have the same location category as the input location `queryLocId` inside the above mentioned rectangular region.
- List containing the following statistical results for the distances found above in the order mentioned:
    - Average of all the distances found in the third output.
    - Standard deviation of the all the distances found in the third output.

Example 1:

Download the `Locations.csv` file from the folder of Project 1 on LMS or Moodle. An Example interactions are given below:

```
>>> locList,simLocList,distSorted,avgstd = main("Locations-sample-Project1.csv", "L83", 1.5, 2.2)
```

The output variables returnd are:

```
>>> locList
['L3', 'L18', 'L47', 'L53', 'L91']
>>> simLocList
['L3', 'L18', 'L53']
>>> distSorted
[0.4804, 1.6936, 2.2567]
>>> avgstd
[1.4769, 0.7412]
```

Explanation:

In Figure 1, the search space using the above mentioned input arguments is shown. Here, the location pin point in red is the query location, i.e., `queryLocId = "L83"`, `d1 = 2.2` and `d2 = 1.5`. The output `locList = ['L3', 'L18', 'L47', 'L53', 'L91']` are shown in blue locations points within the region. Nevertheless, `simLocList = ['L3', 'L18', 'L53']` contains three locations which has similar category as `queryLocId = "L83"` (Check the category column in `Locations.csv` corresponding to the locations `['L3', 'L18', 'L53']` and `"L83"`.

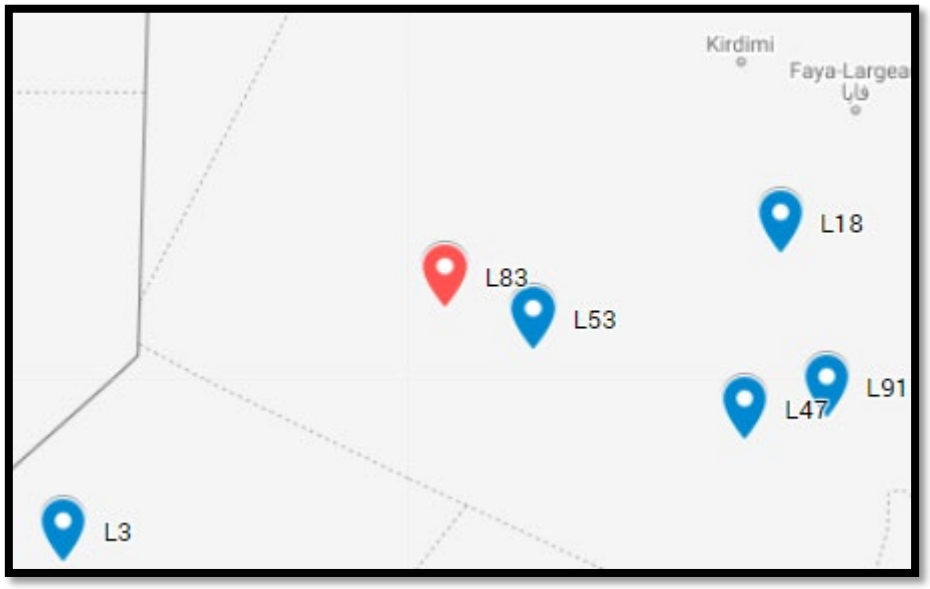

Figure 1: Region constructed using the inputs: `queryLocId="L83", d1=1.5, d2=2.2`

Example 2:

```
>>> locList2,simLocList2,distSorted2,avgstd2 = main ("Locations-
sample-Project1.csv", "L77", 3, 3.85)
```

The output variables returned are:

```
>>> locList2

['L4', 'L15', 'L17', 'L26', 'L30', 'L31', 'L33', 'L38', 'L52',
'L58', 'L88']

>>> simLocList2

['L17', 'L52']

>>> distSorted2

[3.2559, 4.3635]

>>> avgstd2

[3.8097, 0.5538]
```

Assumptions:

Your program can assume the bellow assumptions:

- Anything that is meant to be a string (i.e. header row) will be a string, anything that is meant to be a neumeric will be numeric.
- The string data needs to be considered as case insensitive for input parameter as well as inside the file. For instance, the location id "L35" can be provided as "l35".
- The order of columns in each row will follow the order of the headings provided in the first row. The rows may be in random order and their number are not constant.
- The `main()` function will always be provided with valid input parameters.
- The formula to calculate distance and standard deviation can be found at the end of the project sheet.

Important grading instruction:

You will have noticed that you have not been asked to write specific functions. That has been left to you. However, it is important that your program must defines the top-level function main(). The idea is that within main(), the program calls the other functions. (Of course, these may call further functions.) The reason this is important is that when your program is tested, the testing program will call your main() function. So, if you fail to define main(), my testing program will not be able to test your program and your submission will be graded zero. Don't forget the submission guidelines provided at the start of the project sheet.

Things to avoid:

There are a few things for your program to avoid.

- You are not allowed to import any Python module. While use of the many of these modules, e.g. csv or math is a perfectly sensible thing to do in a production setting, it takes away much of the point of different aspects of the project, which is about getting practice opening text files, processing text file data, and use of basic Python structures, in this case lists and loops.
- Do not assume that the input file names will end in .csv. File name suffixes such as .csv and .txt are not mandatory in systems other than Microsoft Windows.
- Ensure your program does NOT call the input() or print() functions at any time. That will cause your program to hang, waiting for input that automated testing system will not

provide. In fact, what will happen is that the marking program detects the call(s), and will not test your code at all which may result in zero grade.

Submission:

The name of the file containing your code should be your student ID e.g. **12345678.py**. Submit your solution before the deadline electronically on Moodle. No other method of submission is allowed. Your program will be automatically run on Moodle for sample test cases provided in the project sheet if you click the "check" link. However, your submission will be tested thoroughly for grading purpose by teaching team after the due date. Remember you need to submit the program as a single file and copy-paste the same program in the provided text box. You have only one attempt to make the submission so don't submit if you are not satisfied with your attempt. You are encouraged to keep your attempt open as there is no way in the system to open the closed submission and reverse your submission. All open submissions at the time of deadline will be automatically submitted. Separate submission system with slight different guidelines will be made available for submissions after due date for special consideration or late submissions.

You need to contact unit coordinator if you have special considerations or making submission after the mentioned due date.

Marking Rubric:

Your program will be marked out of 30 (later scaled to be out of 10% of the final mark).

22 out of 30 marks will be awarded based on how well your program completes a number of tests, reflecting normal use of the program, and also how the program handles various states such as different number of rows in the input file or input file missing data, etc. You need to think creatively what your program may face. Your submission will be graded by data files other than the provided data file. Therefore you need to be creative to look into corner or worst cases. I have provided few guidelines from ACS Accreditation manual at the end of the project sheet which will help you to understand the expectations.

8 out of 30 marks will be awarded on *style* (5/8) "the code is clear to read" and *efficiency* (3/8) "your program is well constructed and runs efficiently". For style, think about use of comments, sensible variable names, your name at the top of the program, etc. (Please watch lectures, where this is discussed.)

Style Rubric:

0	Gibberish, impossible to understand
1-2	Style is really poor or fair
3-4	Style is good or very good, with small lapses
5	Excellent style, really easy to read and follow

Your program will be traversing text files of various sizes (possibly including large csv files) so try to minimise the number of times your program looks at the same data items.

Efficiency Rubric:

0	Code too incomplete to judge efficiency, or wrong problem tackled
1	Very poor efficiency, additional loops, inappropriate data reading or use of readline()
2	Acceptable or good efficiency with some lapses
3	Excellent efficiency, should have no problem on large files, etc.

Note: Automated testing is being used so that all submitted programs are being tested the same way. Sometimes it happens that there is one mistake in the program that means that no tests are passed. If the marker is able to spot the cause and fix it readily, then they are allowed to do that and your - now fixed - program will score whatever it scores from the tests, minus 4 marks, because other students will not have had the benefit of marker intervention. Still, that's way better than getting zero. On the other hand, if the bug is too hard to fix, the marker needs to move on to other submissions. Remember, no extra fixes will be accepted after submission.

Formula:

Standard deviation is mathematically expressed as:

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

σ = population standard deviation

N = the size of the population

x_i = each value from the population

μ = the population mean

You can find more details at https://en.wikipedia.org/wiki/Standard_deviation

The Distance (d) between two points A = (x_1, y_1) and B = (x_2, y_2) in cartesian plain is calculated as,

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

In this project, a location point is considered as A = (longitude, latitude).

Extract from Australian Computing Society Accreditation manual 2019:

As per Seoul Accord section D,

A complex computing problem will normally have some or all of the following criteria:

- involves wide-ranging or conflicting technical, computing, and other issues;
- has no obvious solution, and requires conceptual thinking and innovative analysis to formulate suitable abstract models;
- a solution requires the use of in-depth computing or domain knowledge and an analytical approach that is based on well-founded principles;
- involves infrequently-encountered issues;
- is outside problems encompassed by standards and standard practice for professional computing;
- involves diverse groups of stakeholders with widely varying needs;
- has significant consequences in a range of contexts;
- is a high-level problem possibly including many component parts or sub-problems;
- identification of a requirement or the cause of a problem is ill defined or unknown.