CITS2401 Computer Analysis and Visualization

Summer Semester 2017-2018

Assignment 2

Due date: **25 January 2018 @ 5:00PM**

Total marks: 20

This assignment is to be completed in Python and submitted via LMS.

Assignment submission instructions:

- Submit 1 python file containing all your solution via the CITS2401 page on LMS (http://www.lms.uwa.edu.au) in Weekly Modules -> Day 8 -> Assignment 2
- You have unlimited attempts to upload your solution. However, only last submission will be graded.
- Name the file as: *YourSurname_Studentnumber.py*, e.g. Smith_902787.py
- Ensure your solution works on MCL computers because it will be graded on MCL computers using Thonny.
- Failure to follow the submission guidelines may result in the award of zero grade.
- Once you submit your assignment, download it again to ensure that it is uploaded correctly.
- If you are not sure, ask a lab demonstrator to help you with the submission of your assignment.

Plagiarism

All work submitted should be your own. I am sure you will agree that this is for your own good!! Note that we have ways to detect plagiarism in code. For example, there are many possible ways to solve the questions and if your solutions match closely, we will be able to identify this. You must not share your solution with others. Incidences of plagiarism will be taken seriously and will be reported to the Head of School as Academic Misconduct and dealt with as per the university policy.

Assignment Overview

There is one problem which has two major parts. You need to submit your solution as per guidelines mentioned above. Your solution should be generic and will be tested with different data in the same format.

Problem: Remote-control cars have been around for a long time now and with the advent of self-driving technologies even the simplest of toys will likely possess some kind of automation. One such manufacturer has developed a remote-control car which can execute pre-determined programs. However, it is not perfect and will sometimes perform an incorrect action.

This manufacturer when testing their latest development maintain a copy of the intended program but also track what movements have actually been made externally. Each car can perform the following actions:

- Move North
- Move West
- Move East
- Move South

The manufacturer has performed a series of trials where multiple cars have been tested in sequence and have bundled all the data into two csv files per car. One contains the instructions given to each car and other contains the actions performed by each car.

Your task, given both files is to find:

- The final displacement of each RC-car from the origin (in both the x and y axes)
- The total distance travelled by each RC-car

Part 1: Write a function rcCar which accepts three inputs. The first two inputs are two lists of strings containing csv files locations: the first list contains the names of the files (including locations if required) containing the instructions for each car and, the second list contains the names of the files (including locations if required) containing the actual actions. Finally, the function will accept a third input which is the speed argument in m/s indicating the speed of each RC-car. You can assume that all cars will always move at the same speed at all times.

The two csv files will contain lines of the following format: Action, time - where each action will be specified as:

- N, t = Move North for t seconds
- E, t = Move East for t seconds
- W, t = Move West for t seconds
- S, t = Move South for t seconds

The function will return six lists:

- The expected x-displacements for each car
- The expected y-displacements for each car
- The actual x-displacements for each car
- The actual y-displacements for each car
- The expected distances travelled by each car
- The actual distances travelled by each car

All displacements and distances are to be presented in meters and rounded to 2 decimal places for the final returned lists.

Additionally the data will adhere to the following constraints:

- There will never be a 'missing' instruction. The number of expected and actual instructions will always match.
- Both files will contain valid instructions only.

Part 2: In addition to the function described above, write a second function plotRC which will accept the same arguments and perform the same calculations as before but will additionally create the following plots:

- A bar-plot comparing the expected and actual distance covered by each car
- A scatter-plot of the expected final horizontal and vertical displacements for each car
- A scatter-plot of the actual final horizontal and vertical displacements for each car

We expect these plots to be generated as sub-plots and must match the example templates. You can expect that the cars will finally rest between the points (-20,-20) and (20,20). Don't forget the guidelines of plots explained to you during the lectures.

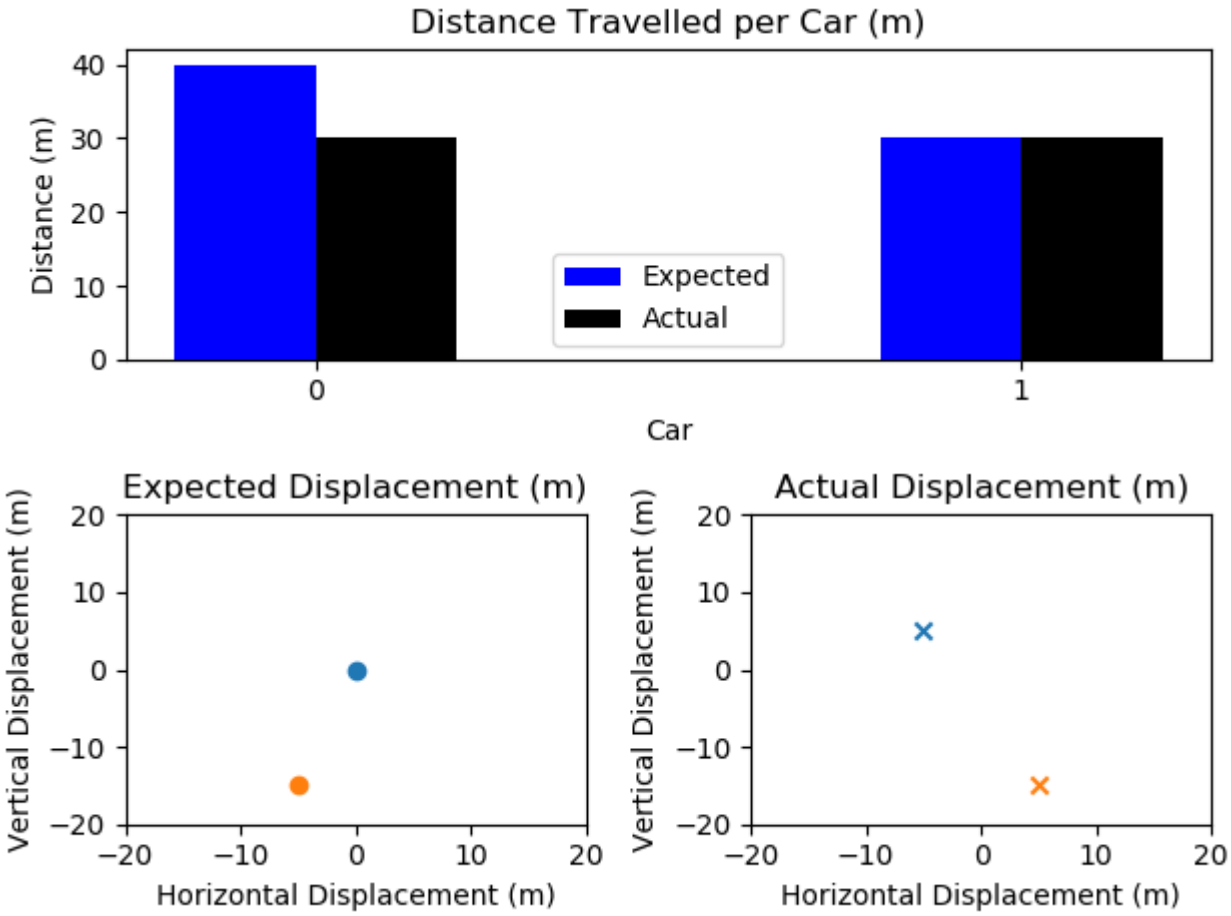
Remember: when writing plotRC don't re-write your code for rcCar, make a function call to that original program to generate your data

Examples and Sample Data:

You can get a sense of what we expect by looking at the following sample data:

- exp1.csv
- exp2.csv
- act1.csv
- act2.csv

You can download the files from LMS from the same location of assignment sheet. The sample plot is:



Hints and Help: Here is some example code on opening and processing a csv file in Python:

```
import CSV

with open('filename', 'r') as file:

    reader = csv.reader(file)

    for row in reader:

        print(row)
```

Where ‘filename’ includes the ‘.csv’ extension ‘r’ indicates you wish to only read the file. There are many other options such as ‘w’ for writing and ‘rw’ for both reading and writing etc.

This assignment will require the use of Sub-plots. To mimic the style of the sample solution use the following general format:

```
plt.subplot(211) #For the bar chart

#Do some plotting

plt.subplot(223) #For the left scatter plot

#Do some plotting

plt.subplot(224) #For the other scatter plot
```

Note: This is a significantly larger assignment than the first, your solution should lie around 100 lines of code and you are suggested to start early.