COMP 636: Python Assessment

Due: **5pm Monday 28 August 2023**

Worth **40%** of COMP636 grade

Submit via Akoraka | Learn

Introduction

The Bankside-Rakaia Motorkhana Mavens (BRMM) car club has asked for a system to help manage its August 'Have-a-go Fun Motorkhana' event. You are provided with an outline of the code for the system to complete.

Motorkhana is a type of motorsport where the aim is to drive around a course made of cones as fast as possible without hitting any cones or going the wrong way.

Drivers in a motorkhana compete individually, on asphalt or on grass. Each attempt ('run') consists of driving around cones, in a particular order (a 'course'), and is timed (see this video). Drivers must memorise a course map (such as Figure 1), then drive the course without hitting any cones. Usually there are 10-20 cones on a course. The winner is the driver with the lowest (fastest) combined times for their runs around the course.

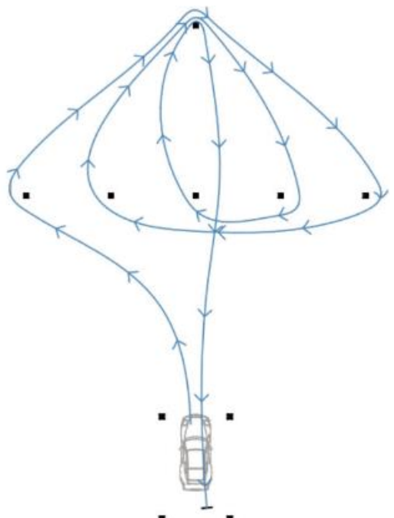
Figure 1: Northland Car Club motorkhana course, www.ncc.org.nz/mk tests.htm

The data provided with this assessment is from the August BRMM 'fun' event. In this event each driver has 3 runs around a course – one run on Course A, one on Course B and one on Course C. Each run has a time recorded. If a driver hits a cone, 5 seconds is added to the driver's time for each cone hit. 10 seconds is added if the driver goes the wrong way around the map ('WD', wrong direction).

For example, in the data, Run 9 is Driver 32 (Hank Barnard) on Course A. He drives the course in 46.22 seconds, but he hits 2 cones and takes a wrong direction. His run total time is therefore 66.22 seconds (46.22 + 5 + 5 + 10):

Run ID	Course	Driver	Time	Cones	WD	**Run Total**
9	A	32	46.22	2	1	**66.22**

For each driver, the *best 2* times from their 3 runs are added together. The winner is the driver with the shortest total time from their best 2 runs. Any driver who completes only two runs has the missing run time ignored. For drivers with less than 2 runs (e.g., Kiri Smith), the result 'HAG' (Had A Go) shows on the overall results instead of a time result.

For example, Hank Barnard's three run totals are 66.22, 47.3 and 63.35 seconds. His result for the day will be his best two, i.e., 47.3 + 63.35 = 110.65 seconds. Luke Cooper did not attempt course B (run 17, no time recorded), so his best two are his course A and C run totals.

Some registered drivers do not have any runs recorded (e.g., 111 Andy Capp, 225 Sonja Pickles). They should show as 'HAG' in the overall results.

Junior drivers (category 'J') are aged 12-19. Junior drivers aged 12-**16** must also have a designated adult caregiver driver (another driver at the event who isn't a junior) registered in the driver list.

Some of the runs have not had their results entered yet for time, number of cones hit and wrong direction (WD), e.g., Driver 220 Andy Pickles did record 3 run times (run numbers 11, 22, 31), but they have not been entered in the provided data file yet. You can use those blank runs for testing your Add/Update Run Data menu option (see below).

Remember that we may use a different set of drivers and runs for marking in brmm_data.py with different data but the same structure. So ***do not hard-code values*** *(such as run 11 or 22, or which times are missing in particular runs, etc.) because we may mark with different drivers and runs.*

File Download:

Download the following files from the COMP636 Assessment block on Akoraka | Learn:

- `brmm_admin_`*`your_name`*`.py` – This is the initial code to begin from.
  Include your **own name** in the filename (e.g., brmm_admin_**Anna_Lee**.py), and your name and student ID in a comment at the start of the file. **Do not change the menu numbering or existing function names**, although you may add new functions of your own.
- `brmm_data.py` – The driver and run data. **Do not change the structure** of this data (col_drivers, etc). Although you may add extra drivers and runs to db_drivers and dbRuns, that is not necessary. We will use our own copy of brmm_data when marking, with different driver and run data, but with the same col_drivers, db_drivers, col_runs, db_runs structure as provided.

Requirements

The system needs to keep a record of drivers, their run results for each course, and the overall results for the day, as well as recording juniors and their nominated parent or caregiver, if appropriate.

- **Drivers** are stored as a dictionary of key:value pairs. Each key is the driver ID. Each value part contains a tuple (First Name, Surname, Category, Age, Caregiver). Category is 'J' for any junior drivers. Drivers who are not juniors do not have an age recorded. Caregiver is the driver ID of an adult driver in the list (for drivers aged 12-16).

- **Runs** are stored as a dictionary of key:value pairs. Each key is a run ID. Each value part contains a tuple (Course, Driver, Time, Cones, WD). Driver is a driver ID. Time is recorded to the nearest 0.01 of a second. Cones contains the number of cones hit, if any. WD (wrong direction) is either 1 (went the wrong direction around one or more cones) or 0 (no wrong direction).

- Missing data values are recorded as None.

- One function (list_drivers) for menu option 1 has already been provided for you. This lists all of the drivers and displays Driver ID, First Name, Last Name and Age.

- You must use the provided **column_output** function for all on-screen display of data (except for the graph of cones hit). You will need to convert your dictionary data into the correct format for this function (the list_drivers function gives an example of how to do this). **Do not modify this function**.

- Validate all user input appropriately. If data of the wrong type is entered by the user, this should be captured without causing the program to crash or any other type of error. Also ensure that only valid values that make sense can be entered.

Tasks

50 marks available in total.

Add the following features to the system: *(Marks shown are an indication and may change a little.)*

1. **Menu enhancement (1 mark):** Modify the code so that the user can enter an upper- *or* lower-case X (i.e., X or x) to exit the program.

2. **Junior driver list (5 marks):** List the junior drivers, in order of increasing age. Combine their full names (first name and surname) in one column (e.g., 'Kiri Smith', not 'Kiri' 'Smith'). For juniors aged 12-16, display their caregiver's full name, not the caregiver's driver ID.

3. **List results of runs (8 marks):** List the run results. For each run, include the time, cones hit and WD status, and then the calculated Run Total time. Ensure your list shows all Course A runs first, then B then C, ordered from best to worst Run Total for each course.

4. **Enter or update run data (15 marks):** First display the run results (from Task 3 above), then allow the user to add or update the times, cones hit and WD, for the existing runs. Ask the user for the ID of the run you want to update and check that the run ID exists. The user can enter the number of cones hit, but 0 is entered automatically if no value is entered. Ensure the time and number of cones is in a sensible range. The user types in 'wd' to enter a wrong direction (in upper or lower case), otherwise no value for WD should default to 0. After updating a run, a user has the option to update another run or return to the menu. (Note: Your code does *not* need to *add* new runs – it just updates existing runs. You can assume that all of the runs are already in the system.)

5. **Display overall results (8 marks):** Calculate and display the overall results for the August 'Have a Go'. For each driver show the sum of their best two runs. Show results from best to worst, with any "HAG" results at the bottom of the list. Show each driver's full name in one column and show '(J)' after the name for junior drivers (e.g., 'Edward Cooper (J)').

6. **Cone graph (8 marks):** Display a graph of the number of cones hit by each driver. Exclude drivers who did not hit any cones. Choose any symbol or character (e.g., Δ, C, ☐) to represent each cone hit, with a space between each symbol or character. For example, using σ as a symbol (you will use a better headings and a better symbol than this!):

```
HEADING1       HEADING2
Joe Bloggs     σ σ σ σ σ
Manu Dodds     σ
Didi Wong      σ σ
```

Additional notes:

- The quality of the user experience will be taken into account, for each of the tasks above. Full marks for any item will require validation of data entered (for data type and sensible values) and details in the interface that demonstrate some consideration of what would work well for the user (within the limitations of the terminal window output in VS Code).

- The provided `brmm_admin_your_name.py` Python file contains a menu structure and partially completed functions. These must not be deleted or renamed, but you may add arguments/parameters to these functions. You may also add additional functions of your own. Remember to rename the file to include your name.

- You are expected to apply problem solving skills to practically solve issues as they arise.

- You must add comments to your code. These do not need to be on every line of code but should be written to be enough detail so that if you came back to the code in 12 months' time that you could quickly work out what the code is doing. The existing list_drivers function gives an example of the level of commenting that is expected.

Submission:

Submit (upload) **only** your main Python .py file for marking: `brmm_admin_`*`your_name`*`.py`. Do not include the brmm_data.py file.

- Submit your file via the submission link on the COMP636 Assessment page.

Important note

This is an **individual** assessment. You must not collaborate or confer with others (e.g., telling others exactly what to do, or how to do it, or sharing code, or using ghost writers, etc), but the discussion of general concepts (e.g., how loops work **in general**, not specific to this assessment) is allowed. You may seek clarification and advice from staff. Ensure you are familiar with the University policies on Academic Integrity (see here).

Indicative mark allocation

(This *indicates* where to spend your time but the marks for each part change a little when marking.)

50 marks available in total:

Item	**Marks available**
Menu enhancement	1
Junior list	8
Results for all runs	10
Add/update run	15
Overall results	8
Cones Graph	8
TOTAL	**50**