# Agile Processes - Group Project

## Project Brief:

Your group/team has been asked to complete an application (partially-completed  
previously) to help the organisers maintain, view and edit these records.  
Your application should offer the following menu of options repeatedly until the user  
chooses to quit the application:

1. Show the results for a race  
2. Add results for a race  
3. Show all competitors by county  
4. Display the podium-places of each race  
5. Display all the race times for one competitor  
6. Show all competitors who have won a race  
7. Show all competitors who have not taken a podium-position in any race.  
8. Quit

### 1 – Show the results for a race

Show the user the list of races; request the user to select a race from the list (using a  
number); system display the runners' ID codes together with their times in minutes  
and seconds.

Display the identification codes for the podium-places of the race.

### 2 – Add results for a race

Ask the user for the race's venue and add it to the races file. Display a list comprising  
competitors’ names from the runners file; ask the user for the race-time for each  
competitor, (0 if they did not run the race). Store the results (ID code and time ) in a  
newly created file e.g. MTU\_Cork.txt if the venue was MTU\_Cork.

### 3 – Show all competitors by county

List all the competitors but organise them alphabetically, by county; counties listed  
alphabetically.

### 4 – Show the podium-places for each race

List the races along with the ids of the podium-places runners of each race. Ensure  
that you employ a nicely formatted table for displaying output.

### 5 - Show all the race times and finishing-positions for a selected competitor

Display a list of runners to choose from. Once the user chooses a runner, display  
runner’s name and ID code. For each race that he/she completed show the race  
name, the competitor’s time and finishing-position in the race. The time to be shown  
in minutes and seconds.

### 6 – Show all competitors who have won a race

List the ID codes and names of those runners who have won at least one race.

### 7 – Show all competitors who have not taken a podium-position in any race.

List the ID codes and names of those runners who have not achieved a 1st , 2nd , or 3rd  
place finish in any race.

### 8 - Quit The application quits.

Note: For this project, podium-places refers to 1st , 2nd , & 3rd places.

## Errors in Partial Code:

### 1. Check Menu Options:

Menu initially doesn’t recognise input as value between 1-8. This is formatted by read\_integer\_between\_numbers().

To fix, **if maximum <= users\_input >= mini:** needs to be changed to **if mini <= users\_input <= maximum:**.

It now checks if the input is greater than or equal to mini and less than or equal to maximum.

Menu only has options 1-7 – missing the “show all competitors who have not taken a podium-position in any race” function. Updated the menu to loop unless 8 is clicked and added the function call for option 7, which will be implemented at a later time.

### 2. Show the results for a race

* Show the user the list of races

It does print the list of races however, the indexing on the list is incorrect. It starts at 0 and adds an extra empty value, i.e. it goes from 0-5 but should be 1-5.  
  
Debugging the code, the races\_location is stored like:   
  
**['Kinsale, 30', 'Blarney, 32', 'Newmarket, 29', 'Youghal, 29.5', 'Castletownbere, 32.5', '']**  
  
races\_location is handled in race\_venues(). Checking the Races.txt file, there is an additional blank line on line 6 which is causing the empty string. It’s also printing the target times.  
  
Adding the code below to race\_venues() prevents splits the code up by comma instead of the newline:  
  
**race\_name = line.strip().split(',')[0]**

race\_results handles the indexing of the list. Adding **+ 1** to the print statement so that I starts at 1 resolves the issue:   
  
**print(f"{i + 1}: {races\_location[i]}")**

* Request the user to select a race from the list (using a number)

This initially returns a “list index out of range” error. In the

We needed to change **split\_line = line.split(",".strip("\n"))** to **split\_line = line.strip("\n").split(",")**

Also added **if line.strip():** to prevent blank lines from effecting the output.

* System display the runners' ID codes, times in minutes and seconds.

No change needed.

* Display the identification codes for the podium-places of the race.

Added a **podium\_position()** function to calculate the podium places.

### 3. Add results for a race

* Ask the user for the race's venue and add it to the races file.
* Display a list comprising competitors’ names from the runners file;
* Ask the user for the race-time for each competitor, (0 if they did not run the race).
* Store the results (ID code and time ) in a newly created file e.g. MTU\_Cork.txt if the venue was MTU\_Cork.

### 4. Show all competitors by county

* List all the competitors
* Organise them alphabetically, by county; counties listed alphabetically.

### 5. Show the podium-places for each race

* List the races along with the ids of the podium-places runners of each race.
* Ensure that you employ a nicely formatted table for displaying output.

### 6. Show all the race times and finishing-positions for a selected competitor

* Display a list of runners to choose from.
* Once the user chooses a runner, display runner’s name and ID code.
* For each race that he/she completed show the race name, the competitor’s time and finishing-position in the race.
* The time to be shown in minutes and seconds.

### 7. Show all competitors who have won a race

* List the ID codes and names of those runners who have won at least one race.

### 8. Show all competitors who have not taken a podium-position in any race.

* List the ID codes and names of those runners who have not achieved a 1st , 2nd , or 3rd  
  place finish in any race.

### 9. Quit The application quits.

* Quit the application if 8 is entered.