# Hints to get to Merit and Excellence grades

## Coding hints

- Don't use CamelCase **and** underscore. Choose one and stick with it.
- Maximum line length needs to be 79.
- Use variable names that clearly say what the variable is.
- Make sure you use numeric, string, dictionary and list coding.
- The user should use keyboard arrows/letters to control the car.
- Use a graphic user interface (GUI), ie Pygame.
- Write to a file for the high score.
- Use a third party or non-core library through Pygame.
- Have blank lines between blocks of code to set it out clearly.
- Variable names must clearly indicate what the variable is for (eg don't use clk, use clock as the variable name).
- Here are a few extra good ideas:
    - check whether the user really wants to quit if they press the X button or Escape key. Including this input validation would allow for handling of invalid input (so you could include this info in your testing and debugging).
    - where possible, use constants, variables and derived values rather than literals, thus making the program more flexible, eg
        - SCREEN_WIDTH and SCREEN_HEIGHT constants to hold the values for the screen dimensions
        - using a derived value to get the centre point of the screen, to display messages
- Use classes and objects, group sprites etc – this would help to make the program more flexible and robust.
- Assign random speeds to each enemy car as this would make the game more engaging/realistic.

- Use FPS constant for frames per second for the game.

- Grouping all ‘enemy car’ sprites allows for more efficient and effective coding (no need to do individual collision detection for each car separately.

```
all_sprites_list = pygame.sprite.Group()
```

## Commenting hints

- Commenting needs to start with a capital letter, and end with a full stop. Treat it like a sentence or paragraph.

- Leave a space after the #.

- It needs to be indented to the same level of the code you are writing about.

- Try and comment on a block or section of code rather than each line.

- Make sure you clearly describe what the code is doing. Don’t describe what is already clear from the code itself, include this and expand on it.

- For merit and excellence you can’t include any redundant comments

## Testing and debugging hints

- Use the testing plan provided as you have a better chance of getting into a higher grade.

- Make sure you show evidence of testing and debugging throughout the creation of your game, not just when it is complete.

- You must test expected, boundary and invalid inputs, and this must be done in an organised way if you wish to get to the higher grades. You need to test one area of functionality at a time, and make any changes needed before testing the next area. Make notes of all errors you identify, don’t rely on memory to recall issues later.

- Here is an example of some testing – it may give you an idea of what you could test.
    - the user presses the expected keys (these are expected inputs)
    - the user presses ‘W’ rather than ‘Q’ when they want to quit (this is a boundary input)
    - the user presses the keys next to the expected keys (these are boundary inputs)

- Here is an example of how to complete the Testing Table (but word yours differently)

**Test Plan (extract)**

Tests for one specific area of functionality

*Test plan: Arrow keys for movement*

| Test (enter) | Output expected | Correct? | Notes |
|---|---|---|---|
| Up arrow pressed | Player car has appearance of accelerating. | ✓ | |
| Right arrow pressed | Player car moves to right. | ✓ | |
| ... | ... | ... | |
| ... | ... | ... | |

Tests for expected cases

Tests for a different area of functionality

*Test plan: Check if want to play again (when game over)*

| | | | |
|---|---|---|---|
| S key pressed (instead of A) – to continue playing | "Play again" message continues to be displayed. | ✓ | |
| W key pressed (instead of Q) – to quit | "Play again" message continues to be displayed. | ✓ | |
| ... | ... | ... | ... |
| ... | ... | ... | ... |

Tests for boundary cases

**Test Plan (extract)**

Tests for one specific area of functionality

*Test plan: Escape key or X button pressed*

| Test (enter) | Output expected | Correct? | Notes |
|---|---|---|---|
| Up arrow pressed | Player car has appearance of accelerating. | ✓ | |
| Right arrow pressed | Player car moves to right. | ✓ | |
| X button pressed | "Do you want to quit (Q), continue (A), reset (R)" – msg displayed | × | *Game crashed. Checked and a typo had caused an error (pygame.QUT changed to pygame.QUIT).* |
| S key pressed (instead of A) – to continue playing | Above message continues to be displayed | ✓ | |
| W key pressed (instead of Q) – to quit | Above message continues to be displayed | ✓ | |
| ... | ... | ... | ... |
| User presses Space bar to continue | "Do you want to quit" message displayed. | ✓ | |

Tests for expected cases

Test for invalid case

Tests for boundary cases

Test for invalid case