# Amy Loughman J123978

# Project: Instructions

**Assessment Resources:**

|  |
| --- |
| Marking key available for lecturer via Blackboard.  Students may refer to the lecture material (GitHub and Blackboard) in formulating their answers.  Refer to <https://github.com/NM-TAFE/ipriot-nms-org-template/tree/main/org> for relevant code style guides, git usage, and other organizational policies and procedures related to the scenario |

**Assessment Instructions:**

|  |
| --- |
| Students must complete every section. Answer succinctly using full sentences. At most three paragraphs are expected per answer.  All answers must be at the student’s own words – copying generated code or answers from ChatGPT or other AI tools is **strictly** prohibited.  Please ensure that all instructions are followed **carefully**, and submissions are well-organized, clearly labelled, and accompanied by any necessary explanations or justifications. |

## Objectives

By completing this task, you will demonstrate the following competencies as outlined in ICTPRG430:

* **Modularity**: Implementing the program using a modular approach.
* **Data Structures**: Utilizing arrays of primitive data types within a class.
* **File Operations**: Reading from and writing to a text file.
* **Class Design:** Developing multiple classes in response to a specification
* **Object Aggregation**: Employing user-defined object aggregation within a class.
* **Polymorphism**: Implementing polymorphism to enhance code extensibility.
* **Debugging**: Utilizing a debugging tool to troubleshoot your code.
* **Code and Documentation Conventions**: Applying specified coding and documentation standards.
* **Unit Testing:** Conducting and documenting unit tests.

As part of this assessment, you will demonstrate competencies in using a version control system, as outlined in ICTICT449. You will plan, install, create, and manage a repository to control versions of your code for the system you implement to complete the scenario. You must follow the commit and branching standards outlined in the **NMS onboarding guide.**

## Scenario

The City of Moondalup is progressively embracing smart city initiatives to enhance urban living, improve efficiency in city services, and promote sustainable practices. As part of this initiative, the city council is eager to transition to a smart parking solution to optimize carpark usage, reduce traffic congestion, and enhance the overall parking experience for residents and visitors.

You have been contracted to create a prototype solution that uses sensors and displays to provide timely information about available parking bays as well as relevant information about weather and other community messages.

The city’s Chief Technology Officer (CTO), has outlined the following requirements:

* The system must accurately track availability of bays in real-time.
* The system must store the scanned license plates of cars
* The display must be updated promptly as cars enter or exit.
* The system should be robust, easy to maintain, and scalable for future enhancements.
* The application must follow best coding practices and include unit tests.
* You must use Git and GitHub for version management.

## Coding requirements

To meet the specifications of the project, you must do the following:

* Create at least three classes
* At least one class must include three or more parameters
* At least one class must *aggregate* another class
* At least one class must include a list (array) of primitive data types
* You should demonstrate an example of polymorphism
* Include at least two unit tests
* Read and write configuration from a file
* Allow carpark to be constructed either from a direct call or via the configuration file (two options for object construction)
* Create a main.py demonstrating the core interaction between instances of your classes
* Use PEP8 throughout your code and docstrings for major functions within your code
* Record and provide evidence of debugging
* Apply at least three different documentation conventions (readme, docstring, comments)

## Version control requirements

* Follow the guidelines in the NMS Onboarding guide
* Create a new repository and configure it with a README, .gitignore, and other essential setup files.
* Initialize your local repository and link it to a remote repository on GitHub.
* Create a branch with your work
* Make initial commits with the basic structure of your Carpark system.
* As you develop the system, commit your changes each time you reach a significant milestone or complete a task.
* Make at least three commits to demonstrate the evolution of your project (please note, if you use the project guide to develop your project, you will be asked to make additional commits as proof-of-work)
* Ensure that any down time or service interruptions that may result from your changes are clearly communicated
* Manage any changes or improvements by committing to the repository with clear, descriptive commit messages.
* Finalize your submission by documenting the final outcome by:
  + (a) pushing your changes to your branch
  + (b) creating a descriptive PR and
  + (c) merging the PR with main
  + (d) pull from the remote main to the local main
  + (e) delete the branch both locally and remotely
  + (f) add an annotated tag and push to git
  + (g) add release documentation

## Key references

Organizational programming style guide: <https://github.com/NM-TAFE/ipriot-nms-org-template/blob/main/org/code-style-guide.md>

Commit message and branching guideline: <https://github.com/NM-TAFE/ipriot-nms-org-template/blob/main/org/git-onboarding-guide.md>

## Questions

1. What is an example of polymorphism in your code? Identify the file and line number and justify your answer. In what way did this example help achieve code extensibility?

An example of polymorphism in the car park project is the Sensors class. Here we have the parent class as an abstract class, then inheriting this abstract class features is the EntrySensor and the ExitSensor. The file is sensors.py and encompasses lines 65-72 and 87-94. They both include the same update\_car\_park() method. Although in EntrySensor it is used to call add\_car() to place a new plate into the car park’s log. Then in ExitSensor it is used to call remove\_car() and removes a plate from the car park’s log. The code can be extended later to add more sensors, like introducing multi-story levels to the car park, you could make out which cars are located where in a building, or even which aisle the cars are in and how many spots are free in that area.

1. What is an example of aggregation in your code? Identify the file and line number and justify your answer

Aggregation is a ‘has-a’ relationship between classes, so you could say that CarPark has a Display class in it when it needs to update the display to contain information about the car park, from lines 46-56 in car\_park.py.

Outline at least one *specific* instance where you performed debugging.   
Ensure the assessor can identify a *concrete* problem that affected you. **Generic answers will not be accepted.**

*What was the issue?*

When making the \_\_init\_\_ method for Display, I mistakenly put the attribute and the value around the wrong way. I had; display\_id = self.display\_id, instead of what it should have been; self.display\_id = display\_id, so the assignment was wrong.

*How did you identify the problem?*

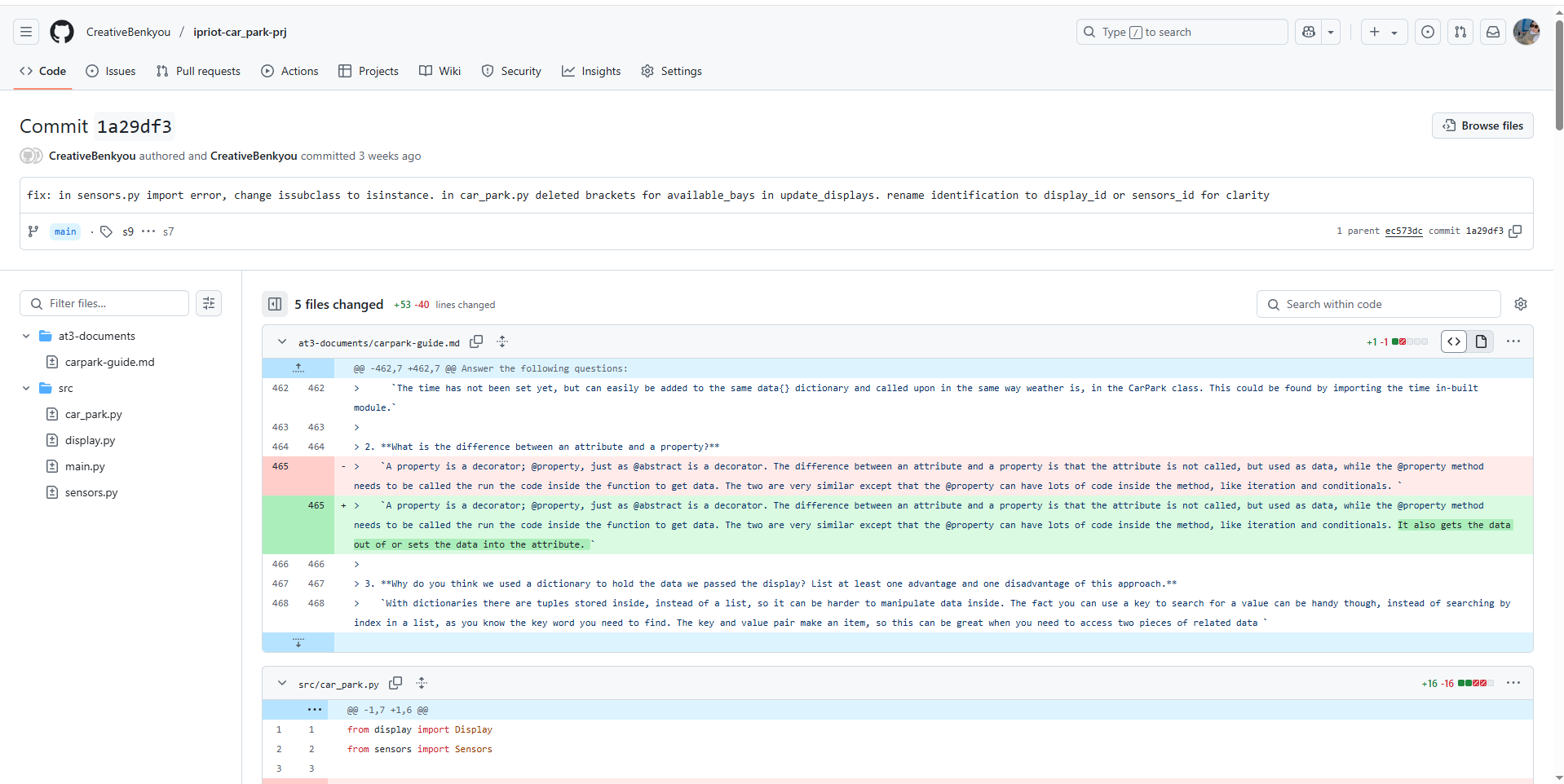
It took me a while of reading the NameError message and reading over the code to spot the error.

*What steps did you take to resolve the problem?*

I was able to track down the error after looking over the code and changing the postion of the attribute.

Attach a screenshot of your GitHub repository that captures part of your development process:

* The screenshot must show the commit graph after an *atomic* commit that isn’t the first or last commit
* The commit message shown must be meaningful and follow the commit guidelines



Provide a screenshot of the PR or release notes you used to document the **final outcome** of the project.

# Coding Checklist

1. Complete the following checklist and reference examples of each of the constructs, where **file name**, is the name of a submitted file, line number, is the line number (or range) for your example.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Item** | **Description** | **File Name** | **Line Number/First Line** |
|  | Class you defined (1) | CarPark | car\_park.py | 11 |
|  | Class you defined (2) | Sensors | sensors.py | 5 |
|  | Class you defined (3) | Display | display.py | 1 |
|  | Reading and Writing from file | To log the activity from cars using their plates as ID in the car park | main.py / moondalup.txt | 7 |
|  | Two options for object construction | main\_car\_park = CarPark()  entry\_sensor = EntrySensor() | main.py | 7  8 |
|  | Class with an array (list or tuple) of primitive (built-in) types | CarPark.plates | car\_park.py | 24 |
|  | Unit test (1) | Arrange: self.entry\_sensor = EntrySensor(car\_park=self.car\_park, display\_id='1', is\_active=True)  Act: Checking if it’s an object of an EntrySensor.  Assert: self.assertIsInstance(self.entry\_sensor, EntrySensor) | test\_sensors.py | 12  15 |
|  | Unit test (2) | Arrange: self.car\_park = CarPark("Perth", 100, 25)  Act: self.car\_park.add\_car("ABC 001")  Assert: self.assertEqual(self.car\_park.plates, ["ABC 001"]) | test\_car\_park.py | 12  24  25 |
|  | Main.py | All classes have been called and new objects created for each. |  | NA |

# Version Control Checklist

1. Complete the following checklist and reference specific commits if applicable

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Item** | **Description/Why** | **Commit Message** | **Commit ID** |
|  | Included .git with submission | *NA* | N*A* | *NA* |
|  | Commit (1) | Initial commit. | [chore: initial commit](https://github.com/CreativeBenkyou/ipriot-car_park-prj/commit/1181d710d17bd11f067a891ec1d872fd88a9c690) | 1181d71 |
|  | Commit (2) | Added the first few different files needed for the program to run. | [feat: add car\_park.py, sensors.py, display.py, add class structure to each file](https://github.com/CreativeBenkyou/ipriot-car_park-prj/commit/7ae0715c2f38127692ea7fc879bd2048b4f4ad76) | 7ae0715 |
|  | Commit (3) | Added the constructor and the attributes for each class and also made two child childs of Sensors, being EntrySensor and ExitSensor. | [feat: add constructor and attributes to the display.py display class and sensors.py sensors class and add child classes of sensors class; entry sensor and exit sensor](https://github.com/CreativeBenkyou/ipriot-car_park-prj/commit/4c9c7d1fc440f889c3c221e92be6bc153ddd67c8) | 4c9c7d1 |
|  | Pushed all work in branch to GitHub |  | NA |  |
|  | Pulled work from remote to local main after PR | NA | *<NA if Fast Forward>* |  |
|  | Delete local branch | *<Name used for local branch>* | NA | NA |

## Submission

Your final submission should include:

* Your local git repo in a zip file.
* Your .git/ **must** be included with your zip file.
* Your .gitignore file should exclude any files that are not required for marking
* Do **not** include ~~venv/~~ in your submission.
* This worksheet, completed with your documentation and optionally the completed carpark project guide (only required for evidencing authenticity).

## Assessment Criteria

You will be assessed on:

* The correct implementation of OOP concepts.
* Code functionality and adherence to the provided specifications.
* Use of object-oriented constructs as specified in the **Coding requirements**
* Quality and clarity of code documentation.
* Successful execution and documentation of unit tests.
* Appropriate commit history and steps as per the core instructions.