# 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. |

**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 logic for one object operation 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 two classes with four instance variables each.
* **Object Construction**: Creating a class that offers two options for object construction.
* **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 two unit test cases.

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 Carpark system. 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.

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 the status of each parking bay in real-time.
* 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 testing.
* 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.
* You should demonstrate an example of polymorphism
* Include at least two unit tests
* 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

## 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.
* Delete the remote branch, and then pull changes into your local main branch and delete your local branch.

## Questions

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

In the Sensor class, the subclass of sensor implements update\_car\_park respectively. This is example of polymorphism.

Sensor.py line 36 and line 45

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

The CarPark class has components/objects. This is example of aggregration and in this case

The car park has diplays and sensor objects.

In car\_pary.py line 16,17, and 18

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?*

Dictionary not being updated correctly, I am not able to retreive the value as expected.

*How did you identify the problem?*

Using breakpoint when debugging, I figured the dictionary was being “overwritten” when my

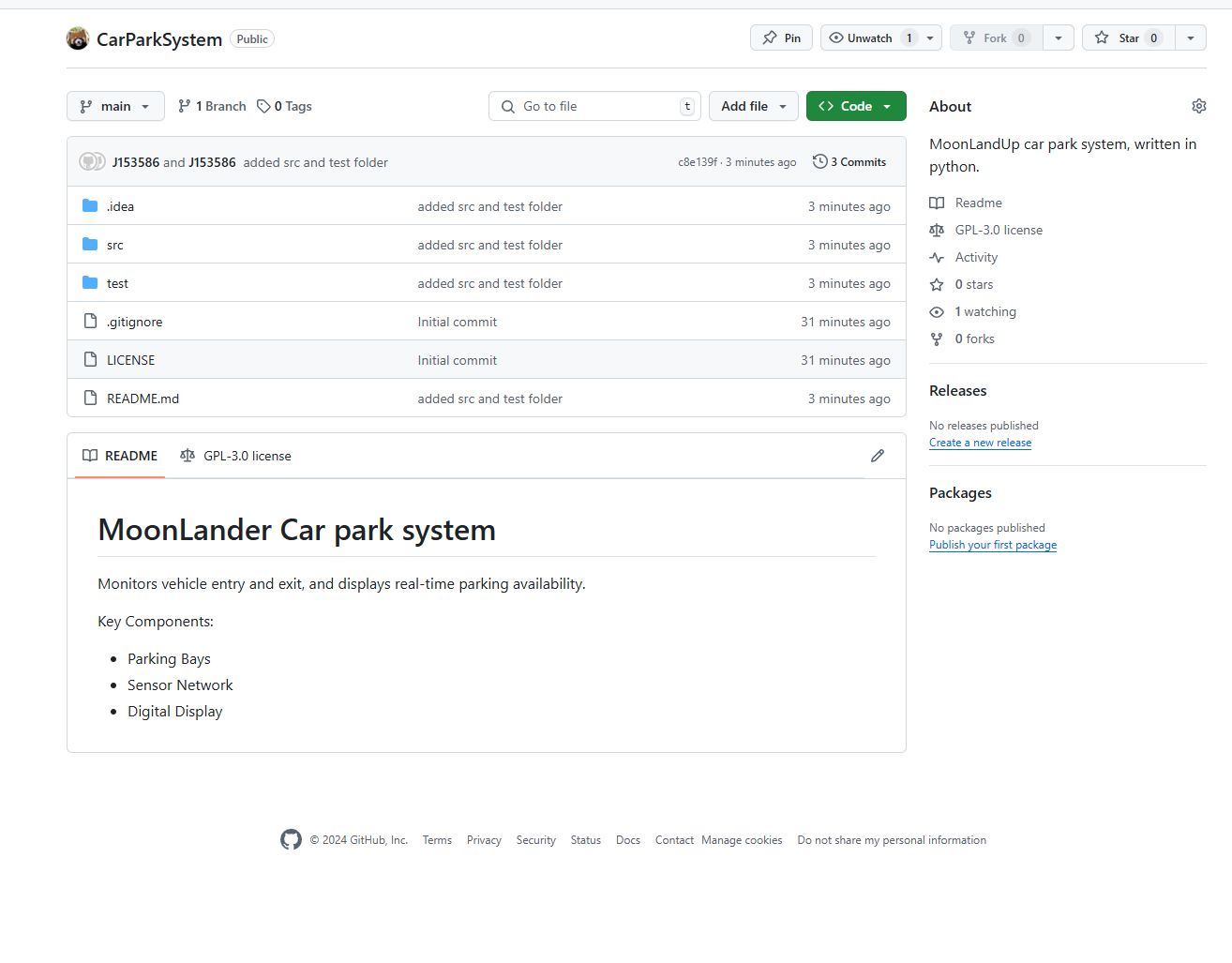
Itention is to add to the dictionary.

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

By researching online, I found the syntax to combine multiple dictionary.

Self.message = {\*\*self.message, \*\*data}

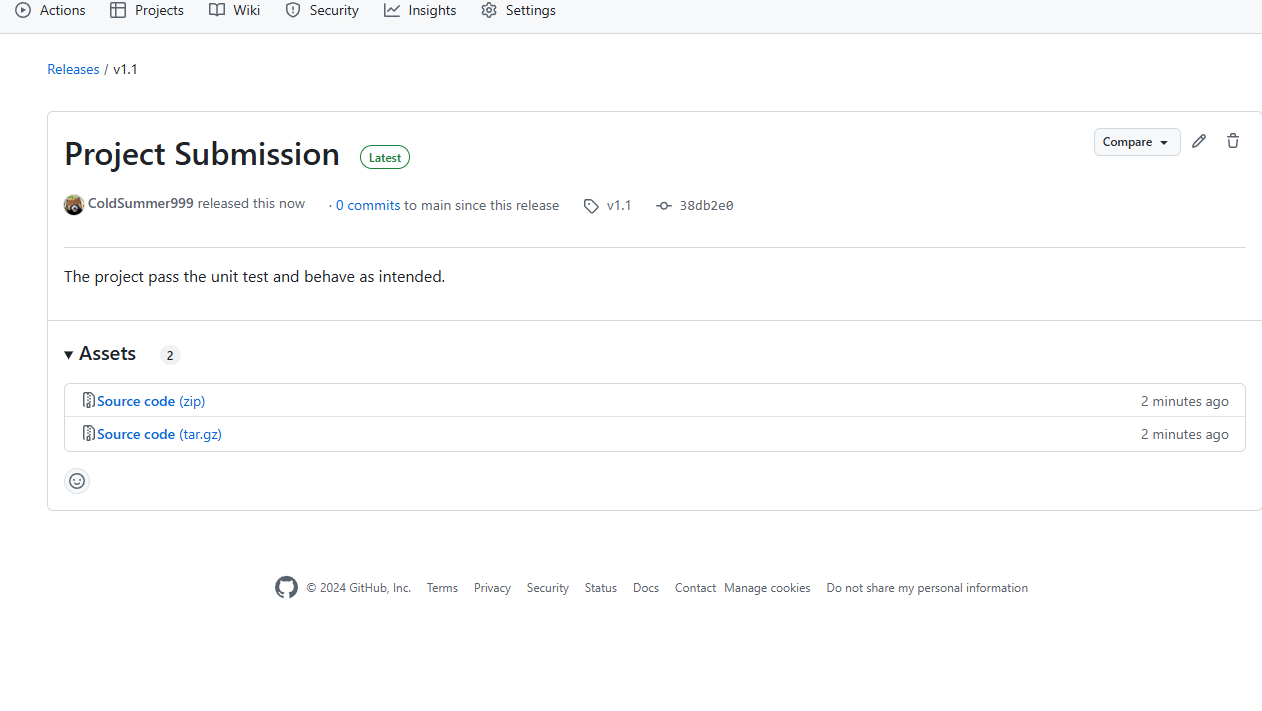
Attach a screenshot of the initial set up of your repository (local and remote)



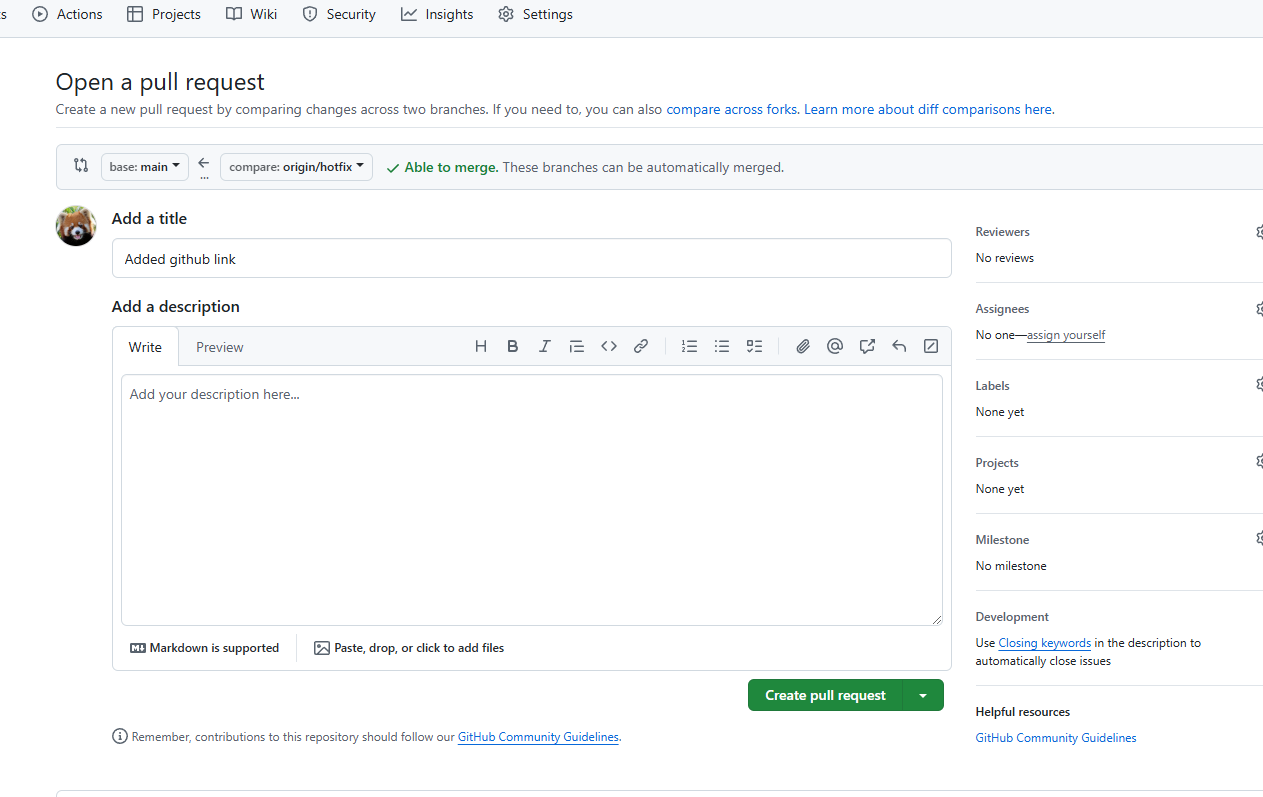
(Did not take a screen shoot setting up the local repo)

But to retrace what I did, I go to the target folder and use the git .init to initialize the local repo. \*Added readme file and git ignore to the folder  
\*git add .  
\*git commit –m “initial repo”  
\*git remote add origin <https://github.com/coldsummer999/carparksystem>\*git push – u origin main

Attach a screenshot of your GitHub repo demonstrating a discrete step in your development process (hint: if you didn’t do it in real time, you can use your revision history!



Provide a screenshot of the PR 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** |
|  | CarPark | Car park object defines the location and capacity of the car park. Has a collection of displays and sensors to help track and display incoming and outgoing vehicles. | car\_park.py | *10* |
|  | Display | Continuously monitor vehicle traffic, capturing license plate numbers upon entry and exit. Transmit this data to the car park management system. | Display.py | 2 |
|  | Sensor | Display a message to incoming and outgoing vehicle. | sensor.py | 6 |
|  | CarPark | Has sensors, displays and plates instances. | car\_park.py | 16,17,18 |
|  | TestSensor | *Testing if the plate attribute is equal to the plate detected by sensor* | test\_sensor.py | 20 - 22 |
|  | TestDisplay | Testing if the message gets updated when using the update method in display | test\_display.py | 21 - 23 |
|  | Main.py | *Instantiate CarPark, sensors, and displays* | Main.py | 22-26 |

# 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* |
|  | Added unit testing | Adds unit testing file for the purpose to evaluate output value. | *Added unit testing* | ed1e2a7ffeac1bd185c4dc2be87338bb5cf66a9f |
|  | Add images to image folder | Evidence of version release on GitHub. | *Added create and release image* | 4784b157466f0db48d8c86ef23f4598955dc422f |
|  | Completed core methods | Core methods in car\_park, sensors and display implemented. | *Core methods completed* | 191767d2674fb35b8f0fc757a2c03d79badb071b |
|  | Pushed all work in branch to github |  | NA |  |
|  | Pulled work from remote to local main after PR | NA | *<NA if FF>* | 34fdf4ecc4ead6aa2dcd603a7402c1544e8baaf1 |
|  | Delete local branch | *Feature/log-car-activity* | 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.

## Assessment Criteria

You will be assessed on:

* The correct implementation of OOP concepts.
* Code functionality and adherence to the provided specifications.
* Quality and clarity of code documentation.
* Successful execution and documentation of unit tests.
* Appropriate commit history and steps as per the core instructions.
* Answers to questions and/or evidencing criteria