PRESENTATION ON PARKING MANAGEMENT SYSTEM

Presentation by:

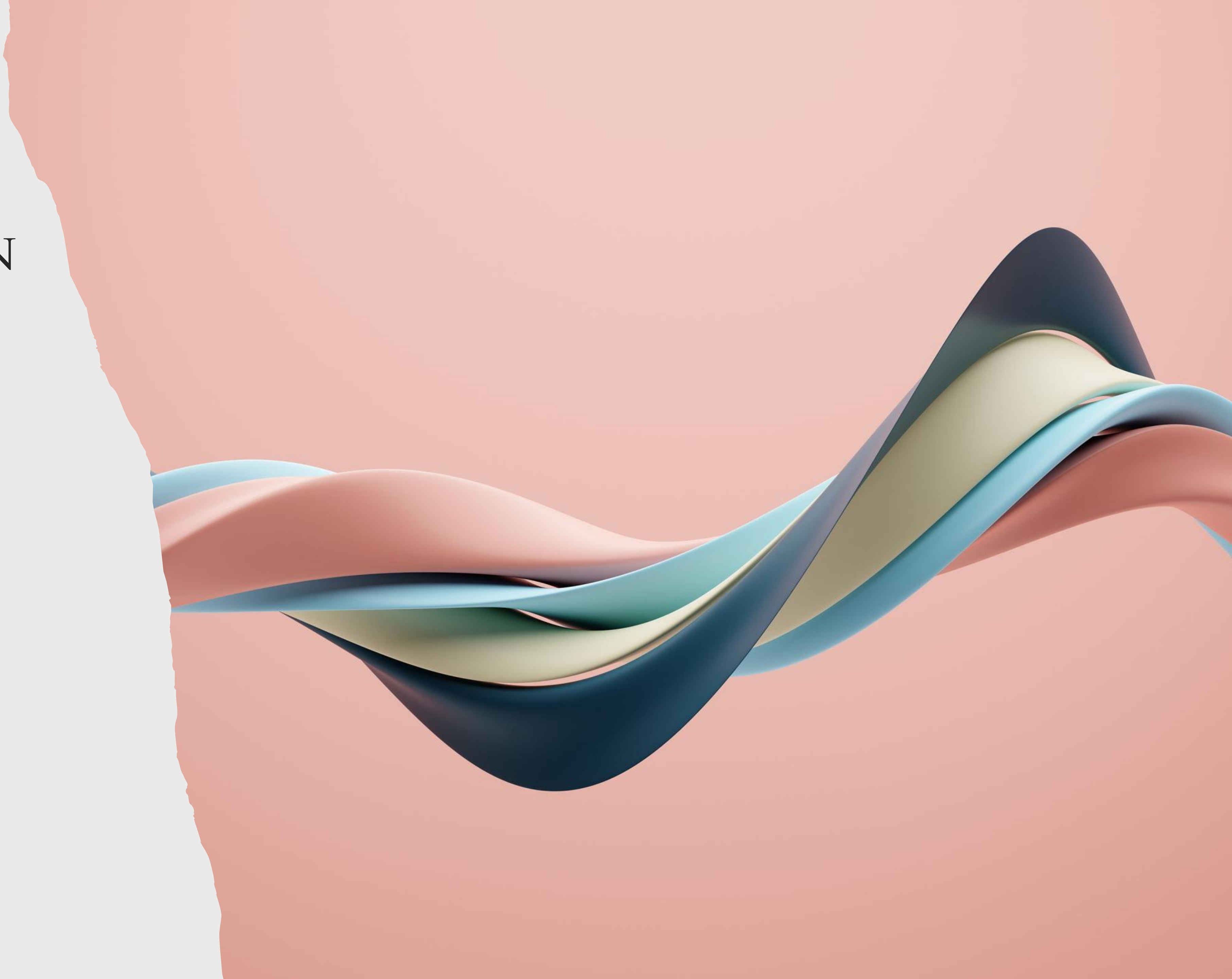
Asrar

Uthresha

Deepak

Eissa

Harsha



INTRODUCTION

- Here's a complete implementation of a Parking Management System in Python, along with the corresponding unit tests using the `unittest` framework.

- This proof-of-concept demonstrates CRUD operations, management of public parking, and tracking parking usage.

EXPLANATION

- **ParkingSpace Class**: Represents an individual parking space, including attributes for whether it's public and occupied.
- **ParkingLot Class**: Manages a collection of parking spaces with methods to add, remove, update, and find spaces. It also includes functions for managing public spaces and tracking usage statistics.
- **Unit Tests**: Implemented to validate that the functionality works as expected. Each method tests a specific aspect of the `ParkingLot` class.

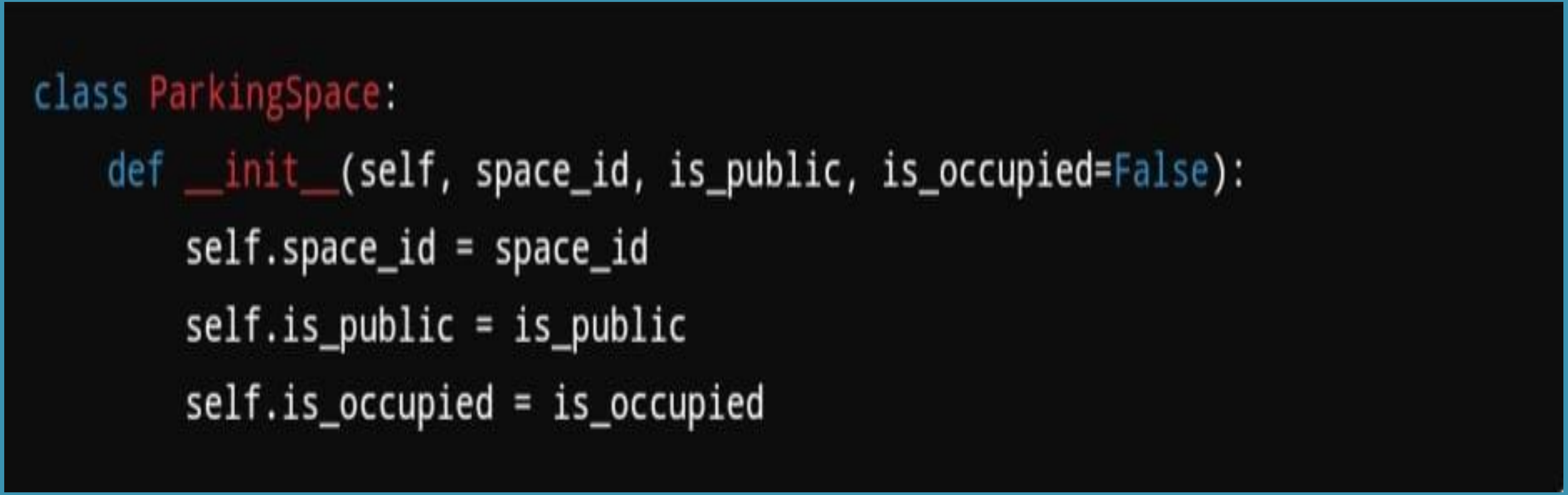
CLASS DEFINATIONS

1 **ParkingSpace** Class:

Purpose: Represents a single parking space.

Attributes:

- **space_id** : Unique identifier for the parking space.
- **is_public** : Boolean indicating if the space is public or private.
- **is_occupied** : Boolean indicating whether the space is currently occupied (default is False).
- python



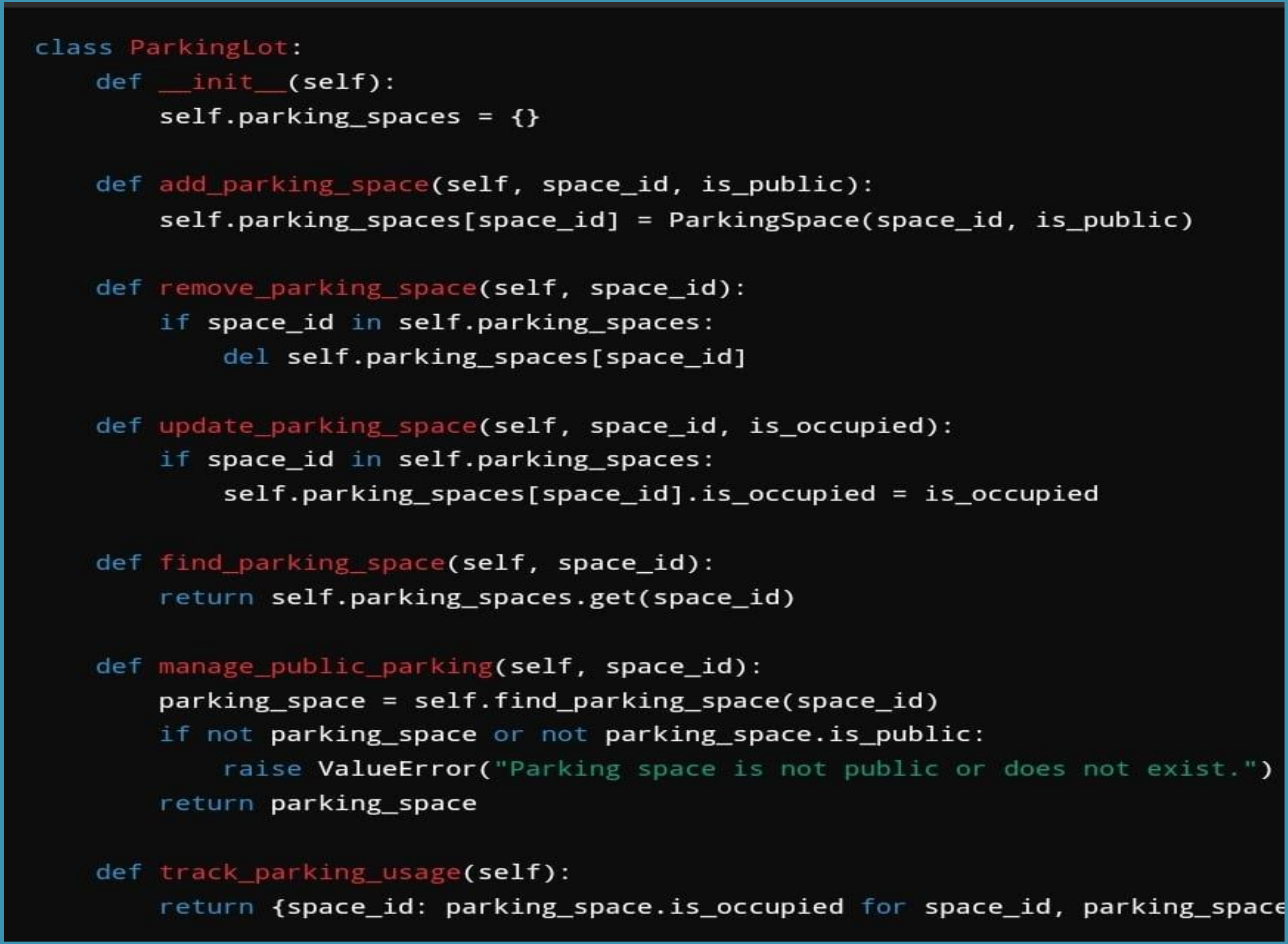
```
class ParkingSpace:
    def __init__(self, space_id, is_public, is_occupied=False):
        self.space_id = space_id
        self.is_public = is_public
        self.is_occupied = is_occupied
```

2 ParkingLot Class:

- **Purpose**: Manages multiple `ParkingSpace` instances.

- Methods:

`add_parking_space`	:	Adds a new parking space.
`remove_parking_space`	:	Removes a parking space by its ID.
`update_parking_space`	:	Updates the occupancy status of a parking space.
`find_parking_space`	:	Retrieves a parking space by its ID.
`manage_public_parking`	:	Ensures that the requested space is public and exists.
`track_parking_usage`	:	Returns a dictionary of the occupancy status of all parking spaces.



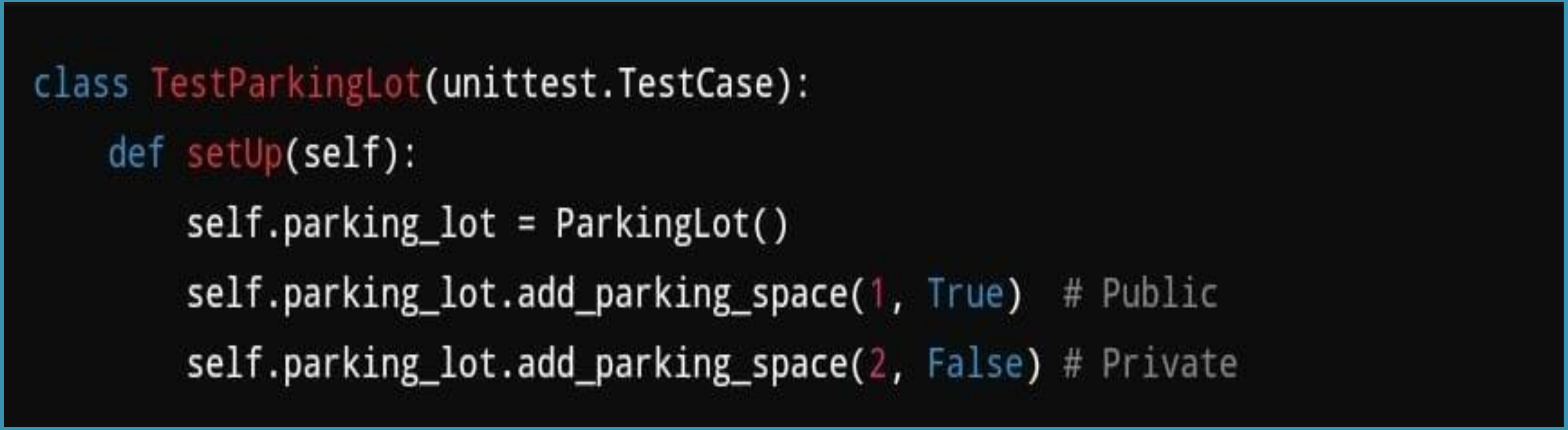
```
class ParkingLot:
    def __init__(self):
        self.parking_spaces = {}

    def add_parking_space(self, space_id, is_public):
        self.parking_spaces[space_id] = ParkingSpace(space_id, is_public)

    def remove_parking_space(self, space_id):
        if space_id in self.parking_spaces:
            del self.parking_spaces[space_id]

    def update_parking_space(self, space_id, is_occupied):
        if space_id in self.parking_spaces:
            self.parking_spaces[space_id].is_occupied = is_occupied

    def find_parking_space(self, space_id):
        return self.parking_spaces.get(space_id)

    def manage_public_parking(self, space_id):
        parking_space = self.find_parking_space(space_id)
        if not parking_space or not parking_space.is_public:
            raise ValueError("Parking space is not public or does not exist.")
        return parking_space

    def track_parking_usage(self):
        return {space_id: parking_space.is_occupied for space_id, parking_space
```

UNIT TESTING

- The code also includes unit tests to validate the functionality of the `ParkingLot` class using the `unittest` framework.
- **`TestParkingLot`** Class:

  This class contains various test methods to check different functionalities of the `ParkingLot` class.

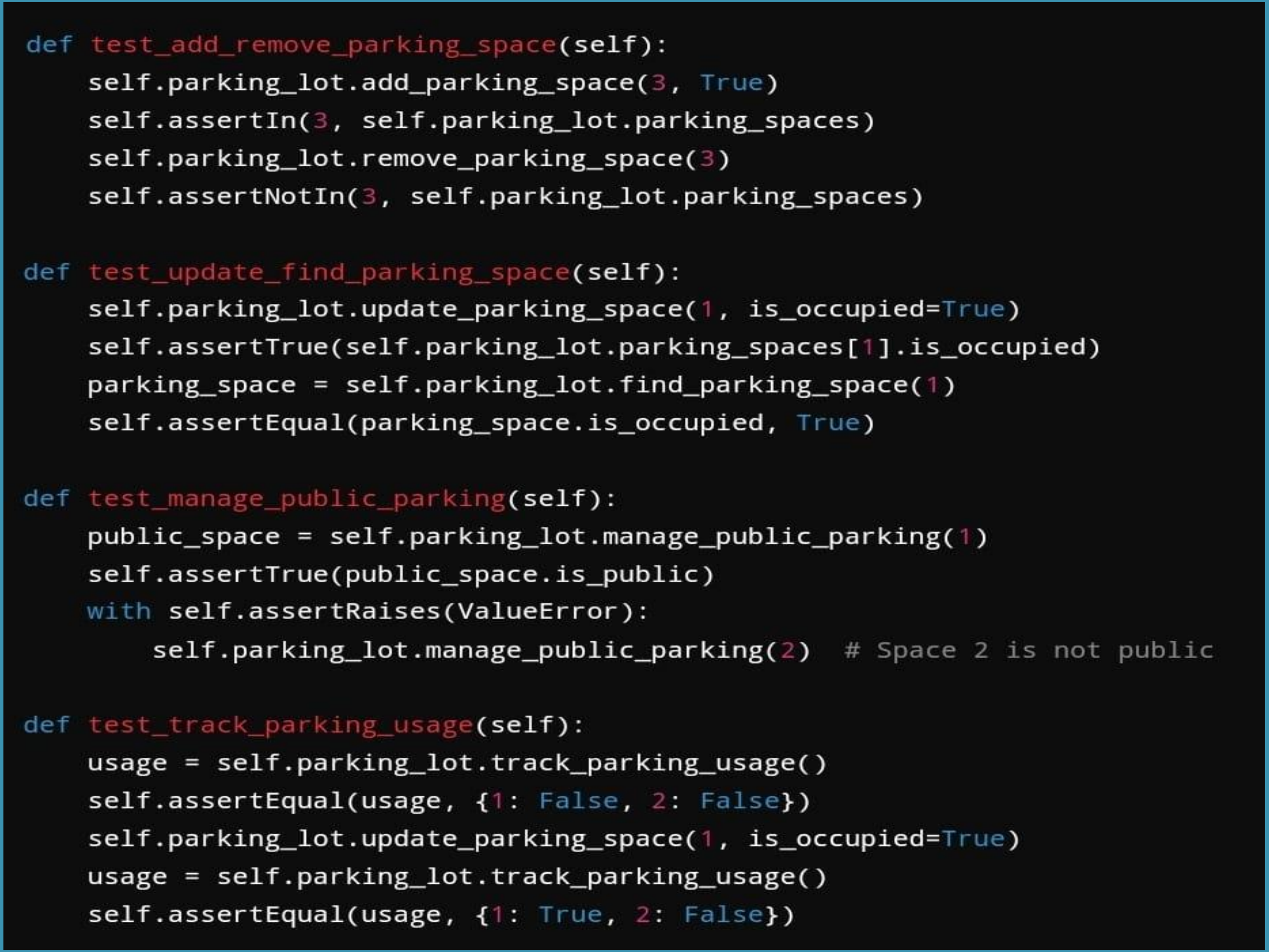
  `setUp` Method: Initializes a new `ParkingLot` instance and adds two parking spaces (one public and one private).



```python
class TestParkingLot(unittest.TestCase):
    def setUp(self):
        self.parking_lot = ParkingLot()
        self.parking_lot.add_parking_space(1, True)  # Public
        self.parking_lot.add_parking_space(2, False) # Private
```

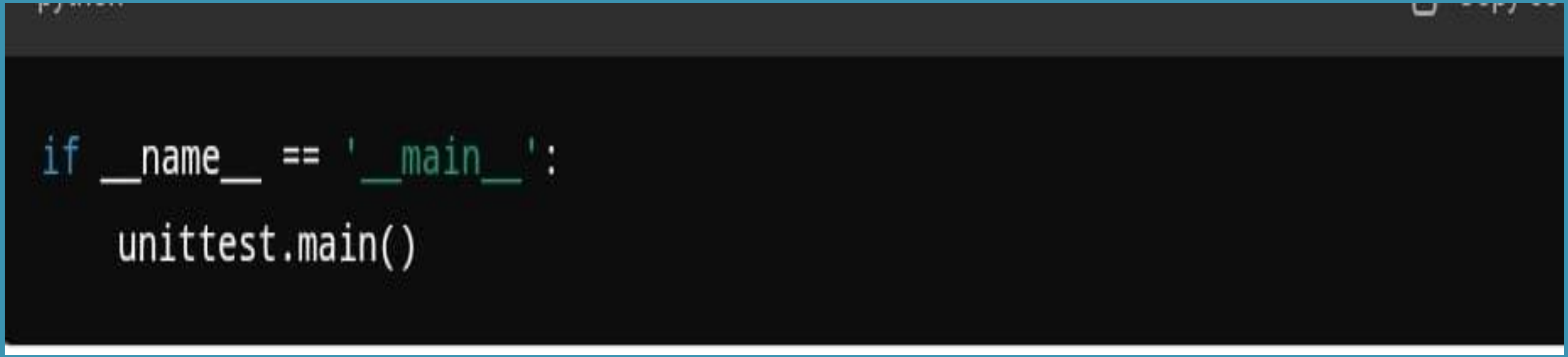
Test Methods:

- **`test_add_remove_parking_space`**:Verifies that parking spaces can be added and removed correctly.
- **`test_update_find_parking_space`**:Checks that updating a parking space's occupancy works and can be retrieved correctly.
- **`test_manage_public_parking`**: Tests the public parking management to ensure it raises an error for non-public spaces.
- **`test_track_parking_usage`**: Validates that the tracking of parking usage reflects the correct occupancy status.



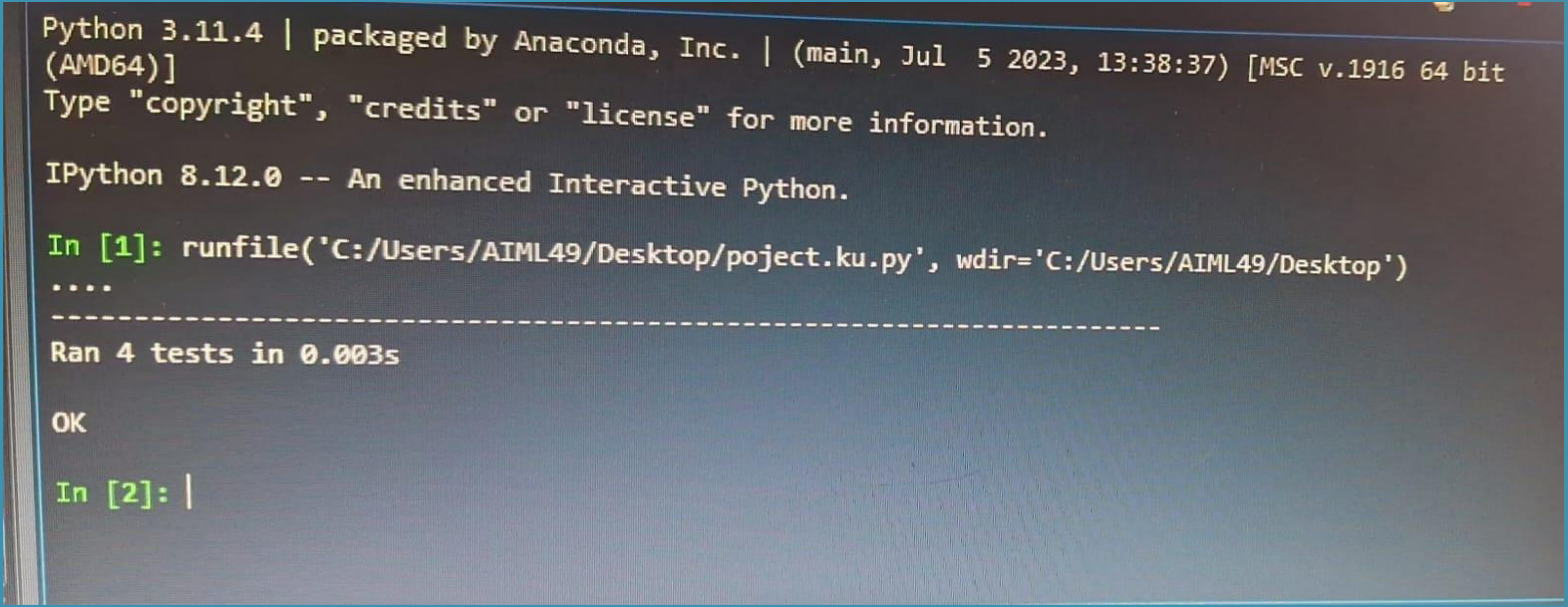
```python
def test_add_remove_parking_space(self):
    self.parking_lot.add_parking_space(3, True)
    self.assertIn(3, self.parking_lot.parking_spaces)
    self.parking_lot.remove_parking_space(3)
    self.assertNotIn(3, self.parking_lot.parking_spaces)

def test_update_find_parking_space(self):
    self.parking_lot.update_parking_space(1, is_occupied=True)
    self.assertTrue(self.parking_lot.parking_spaces[1].is_occupied)
    parking_space = self.parking_lot.find_parking_space(1)
    self.assertEqual(parking_space.is_occupied, True)

def test_manage_public_parking(self):
    public_space = self.parking_lot.manage_public_parking(1)
    self.assertTrue(public_space.is_public)
    with self.assertRaises(ValueError):
        self.parking_lot.manage_public_parking(2)  # Space 2 is not public

def test_track_parking_usage(self):
    usage = self.parking_lot.track_parking_usage()
    self.assertEqual(usage, {1: False, 2: False})
    self.parking_lot.update_parking_space(1, is_occupied=True)
    usage = self.parking_lot.track_parking_usage()
    self.assertEqual(usage, {1: True, 2: False})
```

- Execution:
- If the script is run directly, it will execute the tests.



```python
if __name__ == '__main__':
    unittest.main()
```

- OUTPUT



```
Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul  5 2023, 13:38:37) [MSC v.1916 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.12.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/AIML49/Desktop/poject.ku.py', wdir='C:/Users/AIML49/Desktop')
....
----------------------------------------------------------------------
Ran 4 tests in 0.003s

OK

In [2]: |
```

THANK YOU