Parking Lot System

Parking Lot System Automates the ticketing system to park the cars.

Techstack

- **Programming language**: Python-3.8
  - Dependencies
    - pytest - For running tests
    - pipenv - To isolate and manage virtual environments
- **Operating System**: Ubuntu-18.04 LTS

Installation

If you already have python installed and you don't want to use any virtual environment, just install pytest library

```
pip3 install pytest
```

If you don't want to run tests. You don't have to install anything, you can skip this and can go to 'Usage'

To do fresh installation

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa
```

When prompted press Enter to continue:

```
Press [ENTER] to continue or Ctrl-c to cancel adding it.
```

Once the repository is enabled, install Python 3.8 with:

```
sudo apt install python3.8
```

Verify that the installation was successful by typing:

```
python3.8 --version
```

Install `pipenv` to mange dependencies

```
sudo apt install python3-pip
pip3 install pipenv
cd sr_backend-greenlight_planet
pipenv shell
pipenv install --dev
```

Usage

For use case where input is given as file

```
python3 main.py ./tests/test_data/input.txt
```

For use case where input is entered

```
python3 main.py
```

To run tests

```
python3 -m pytest -vv tests/
```

Folder structure

- **parking_lot**: Main Package where all the logic is implimented
    - **parking_lot.py**: This file contains bussiness logic and can access the the data
    - **run_commands.py**: This file is a driver class which will converts the user commands and call approriate bussiness logic in parking_lot.py
- **tests**: This folder contains all tests related data and units tests
    - **test_data**: This Folder contains test data of various scenatios
    - **test_parking_lot.py**: This test file tests all the methods of class **parking_lot.PrakingLot**
    - **test_run_commads.py**: This test file tests all the methods of class **run_commands.RunCommands**
- **main.py**: This the entry point for user to run the code, this will handle user inputs from shell or from file.
- **Pipfile**: This file just holds dependencies list

Classes explination

parking_lot.py

```
class ParkingLot(builtins.object)
    A class to represent a parking lot give access to manipulate its data.

    ...

    Attributes
    ----------
    _free_slots : int
        free slots available in parking lot
    _max_slots : int
        max slots available in parking lot
    _parking_data : list
        holds car info '{registration_no:"","colour":""}' in a list
    _EMPTY_SLOT : bool
        it is contast value -> 'False', which is used to
        represent empty slot in the _parking_data

    Methods
    -------
    set_max_parking_slots(no_of_slots):
        Intializes the 'Attributes' with given no_of_slots
    get_metadata():
        Returns present values in 'Attributes'
    get_free_slots_count():
        Returns _free_slots
    allocate_slot(car_details):
        Allocates nearest parking slot and returns slot number
    deallocate_slot(slot):
        Remove car from the given slot number
    query(return_key, query_key, condition_value):
        Returns query results based return_key, query_key and conditonal_value
    status():
        returns tab-delimited string of values present _parking_data
```

run_commands.py

```
class RunCommands(builtins.object)
   This class provides abstraction over ParkingLot.
   So, that we can execute commads which are user friendly .

   ...

   Attributes
   ----------
   _parking_obj : ParkingLot
```

```
    Methods
    -------
    run_command(command_input):
        Executes the given the command

    Methods defined here:

    __init__(self)
        Initialize self.  See help(type(self)) for accurate signature.

    run_command(self, command_input)
        Checks the commands and calls the specific function of ParkingLot
        and returns the function result

        Parameters
        ----------
        command_input : list
            command_input[0] -> command
                allowed values:
                     - create_parking_lot
                     - park
                     - leave
                     - status
                     - registration_numbers_for_cars_with_colour
                     - slot_numbers_for_cars_with_colour
                     - slot_number_for_registration_number

        Returns
        -------
        returns ParkingLot function output without any transformation.
```

This is a summary please use below command for detialed info.

```
>>> python
>>> from parking_lot.run_commands import RunCommands
>>> from parking_lot.parking_lot import ParkingLot
>>> help(ParkingLot)
>>> help(RunCommands)
```