General Remarks

This was a great assessment where I got to challenge myself and learn a lot. A lot of my time was spent debugging and implementing the UI in Tkinter, which was new to me.

Challenges

I spent quite a considerable amount of time trying to get the result stated in the example. I even drew the grid and simulated the movement. I finally decided to go with my instincts and if I was wrong, I look forward to being corrected. Find the example in question below:

```
# Example usage of the function
initial_position = (0, 0, 'N')
commands = ['F', 'R', 'F', 'L', 'B']final_position = move_robot(commands, initial_position)
print(final_position)
# Output: (1, 1, 'N')
```

My output, however, is: (1, 0, 'N')

Improvements made

- I also decided to make the UI real time so that the user can move the robot one step at a time and see it move on the grid. I did this to simulate the real world experience where one would want to see if the robot is near the walls of the grid so as to avoid an accident. However, I believe that my move_robot function meets all the requirements and I tried my best to make it readable and optimized. The time complexity of the function is O(n), where n is the number of commands. This is because the function iterates through each command once and performs constant-time operations within each iteration. The space complexity of the function is O(1). It doesn't use any additional data structures whose space requirements depend on the input size. Instead, it relies on a fixed number of variables to store the state of the robot.

- I also made a few unit tests to check whether the algorithm was working as need be

Shortcomings

I hosted the code on github but did not find the time to host the UI so that it can be accessed online.

Opportunities/Areas of Improvement

1. Errors can be handled more graciously, rather than raising them. I was conflicted on whether to raise them or fix them before starting my processing. I wondered whether to be “Gracious with what I accept but strict on what I output” or be strict throughout to avoid reducing the quality of the system in the long run.
2. Hosting of the UI online so that even non-developers can see the simulation
3. Running the code on an actual robot