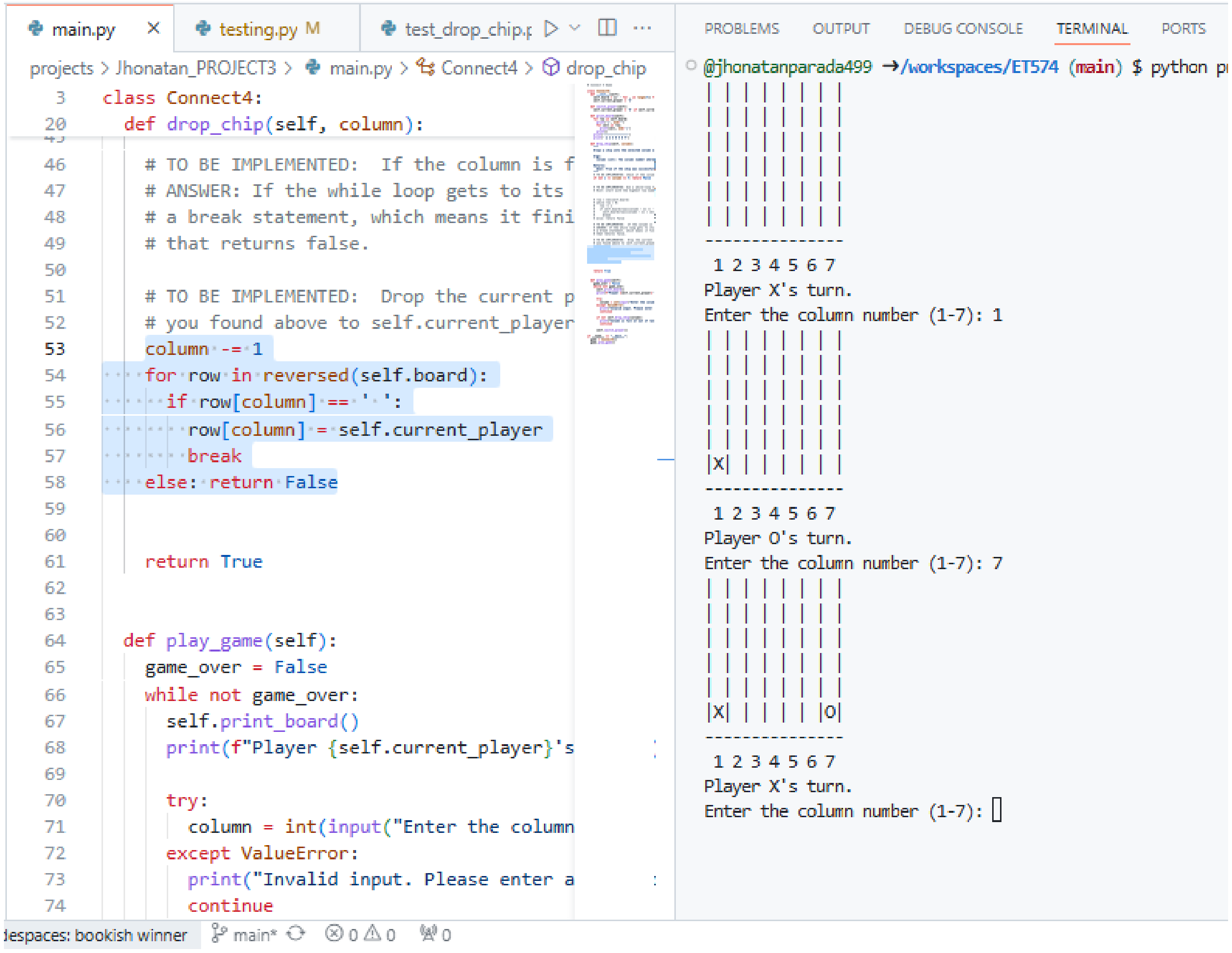
Project 3 (part 1)

Jhonatan Parada

ET574

Main.py



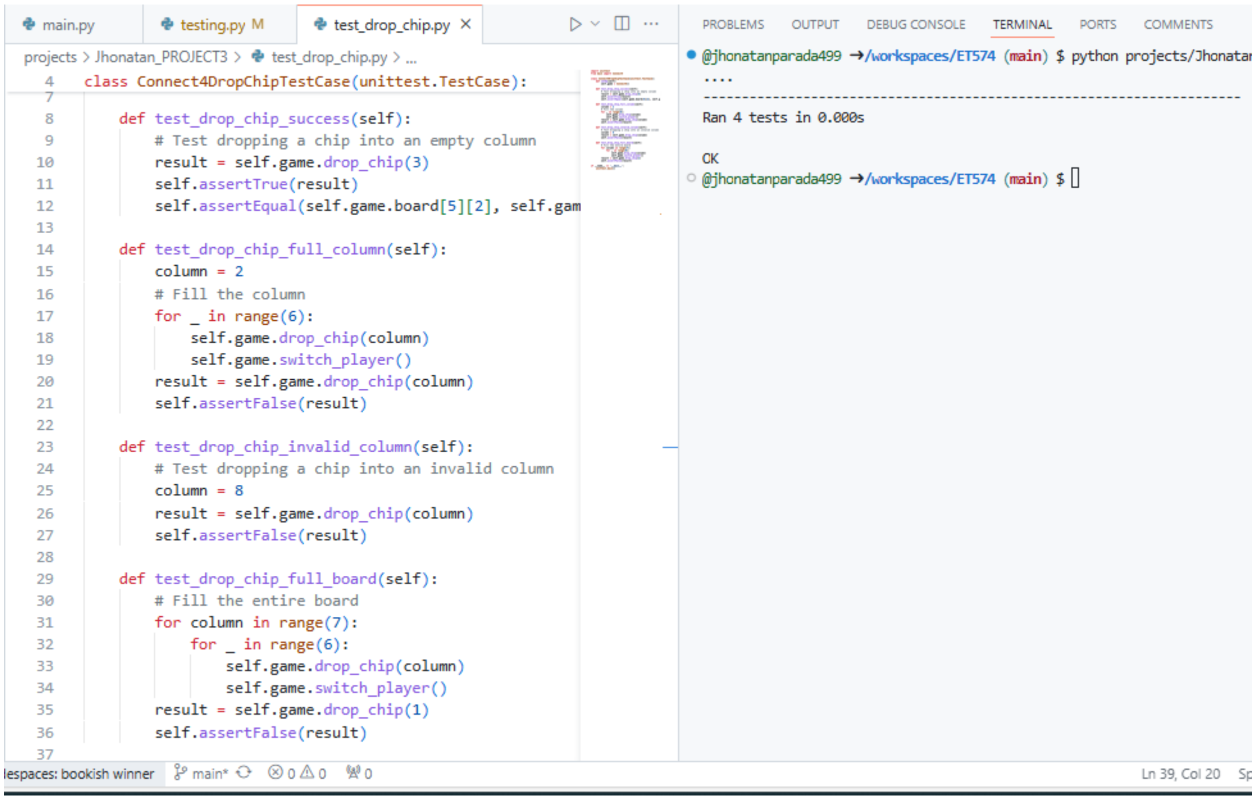
```python
class Connect4:
    def drop_chip(self, column):
        # TO BE IMPLEMENTED:  If the column is f
        # ANSWER: If the while loop gets to its
        # a break statement, which means it fini
        # that returns false.

        # TO BE IMPLEMENTED:  Drop the current p
        # you found above to self.current_player
        column -= 1
        for row in reversed(self.board):
            if row[column] == ' ':
                row[column] = self.current_player
                break
        else: return False


        return True


    def play_game(self):
        game_over = False
        while not game_over:
            self.print_board()
            print(f"Player {self.current_player}'s

            try:
                column = int(input("Enter the column
            except ValueError:
                print("Invalid input. Please enter a
                continue
```

```
@jhonatanparada499 ➜ /workspaces/ET574 (main) $ python p
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
---------------
 1 2 3 4 5 6 7
Player X's turn.
Enter the column number (1-7): 1
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
|X| | | | | | |
---------------
 1 2 3 4 5 6 7
Player O's turn.
Enter the column number (1-7): 7
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
|X| | | | | |O|
---------------
 1 2 3 4 5 6 7
Player X's turn.
Enter the column number (1-7): 
```

Description: I used a for loop because I wanted to write this code implementation with as few lines as possible (Nonetheless, I wrote a version of it using a while loop, but I commented it). The logic of my loop aligns with the same that is instructed in Project_PartI_drop_chip.pdf. So, the first thing I observed is that we are working with two logical ranges, the first one is from 1 to 7, which is the number of columns (user perspective), but from the code side, the number of columns is from 0 to 6, so this has to be fixed or equalized somehow so when a user inputs number 2, the chip will not be displayed in column number 3. That is the reason why I decreased the argument column by one.

Then, I used the reversed constructor to reverse self.board and iterate through its rows logically from bottom to top. What happens next is straightforward, if the column of the current row is empty then we will write the character that represents either player there and break the loop right there so it does not puts more than one chip in each turn and the code will ignore the else part of the for loop, that in case there is no empty slot in the current column, it will return false, indicating that there is not any empty slot.

Test_drop_chip.py



```python
class Connect4DropChipTestCase(unittest.TestCase):

    def test_drop_chip_success(self):
        # Test dropping a chip into an empty column
        result = self.game.drop_chip(3)
        self.assertTrue(result)
        self.assertEqual(self.game.board[5][2], self.gam

    def test_drop_chip_full_column(self):
        column = 2
        # Fill the column
        for _ in range(6):
            self.game.drop_chip(column)
            self.game.switch_player()
        result = self.game.drop_chip(column)
        self.assertFalse(result)

    def test_drop_chip_invalid_column(self):
        # Test dropping a chip into an invalid column
        column = 8
        result = self.game.drop_chip(column)
        self.assertFalse(result)

    def test_drop_chip_full_board(self):
        # Fill the entire board
        for column in range(7):
            for _ in range(6):
                self.game.drop_chip(column)
                self.game.switch_player()
        result = self.game.drop_chip(1)
        self.assertFalse(result)
```

```
@jhonatanparada499 ➜/workspaces/ET574 (main) $ python projects/Jhonatar
....
----------------------------------------------------------------------
Ran 4 tests in 0.000s

OK
@jhonatanparada499 ➜/workspaces/ET574 (main) $ 
```

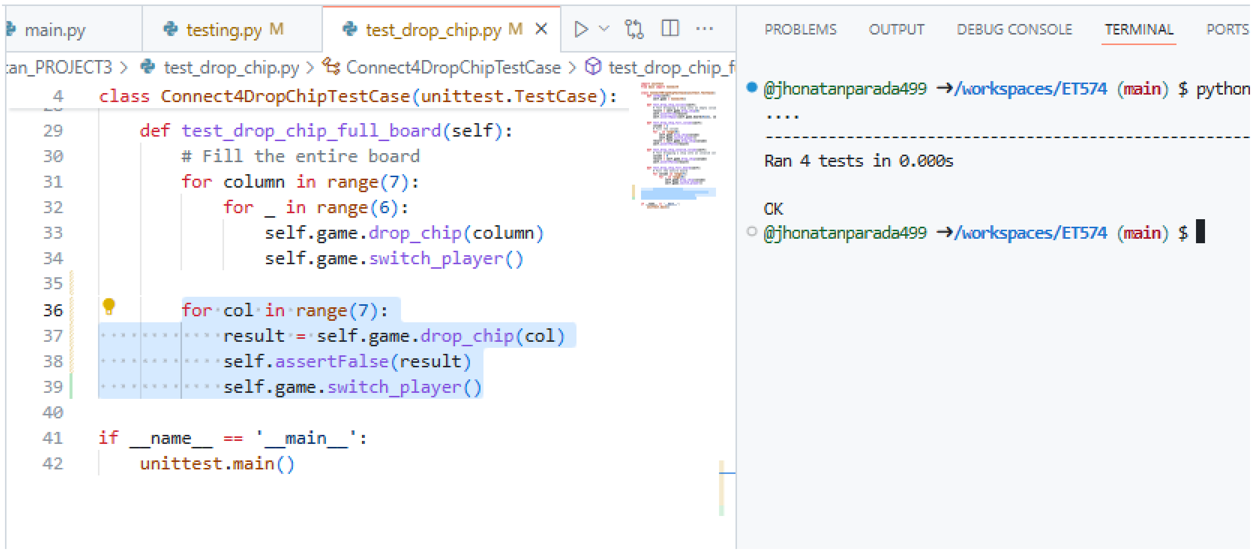
Description: We run the 4 test cases in Test_drop_chip.py and the display shows that everything went ok during the testing.

The first test verifies the success from putting a chip in where the user intended and that the character representing that chip is the same as the logical user that made the logical move.

The second test verifies how the program handles the event of trying to put a chip in a column that is full. In the method drop_chip, we said that if the column was full (which is the same as saying that no character in the logical column is an empty space) then we would return false. Therefore, a False value should be returned in this test case, and we can say it worked as intended because the unittest module is showing no errors in the console.

The third test case is straightforward, the first line of the method drop_chip checks that the input of the user is within the allowed boundaries, in case is not true, it is supposed to return false, and essentially the test case checks that we get false in such scenario where we input a number out of range.

The last test case is like the second test case, but it checks for only number 1 while the other columns are full, yet it is a possible scenario and sometimes the code might behave in ways we don't comprehend so I think it is a good example. I'm going to further test this code by adding additional functionality to this function and check every single column instead of just the first one, they all are supposed to return false, so let's see what happens:



```python
class Connect4DropChipTestCase(unittest.TestCase):
    def test_drop_chip_full_board(self):
        # Fill the entire board
        for column in range(7):
            for _ in range(6):
                self.game.drop_chip(column)
                self.game.switch_player()

        for col in range(7):
            result = self.game.drop_chip(col)
            self.assertFalse(result)
            self.game.switch_player()

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

```
@jhonatanparada499 →/workspaces/ET574 (main) $ python
....
----------------------------------------------------------
Ran 4 tests in 0.000s

OK
@jhonatanparada499 →/workspaces/ET574 (main) $
```

So, everything went as expected this time, I created a loop to simulate users trying to drop a chip in every column of a full board and check that in every case the method drop_chip will return false.