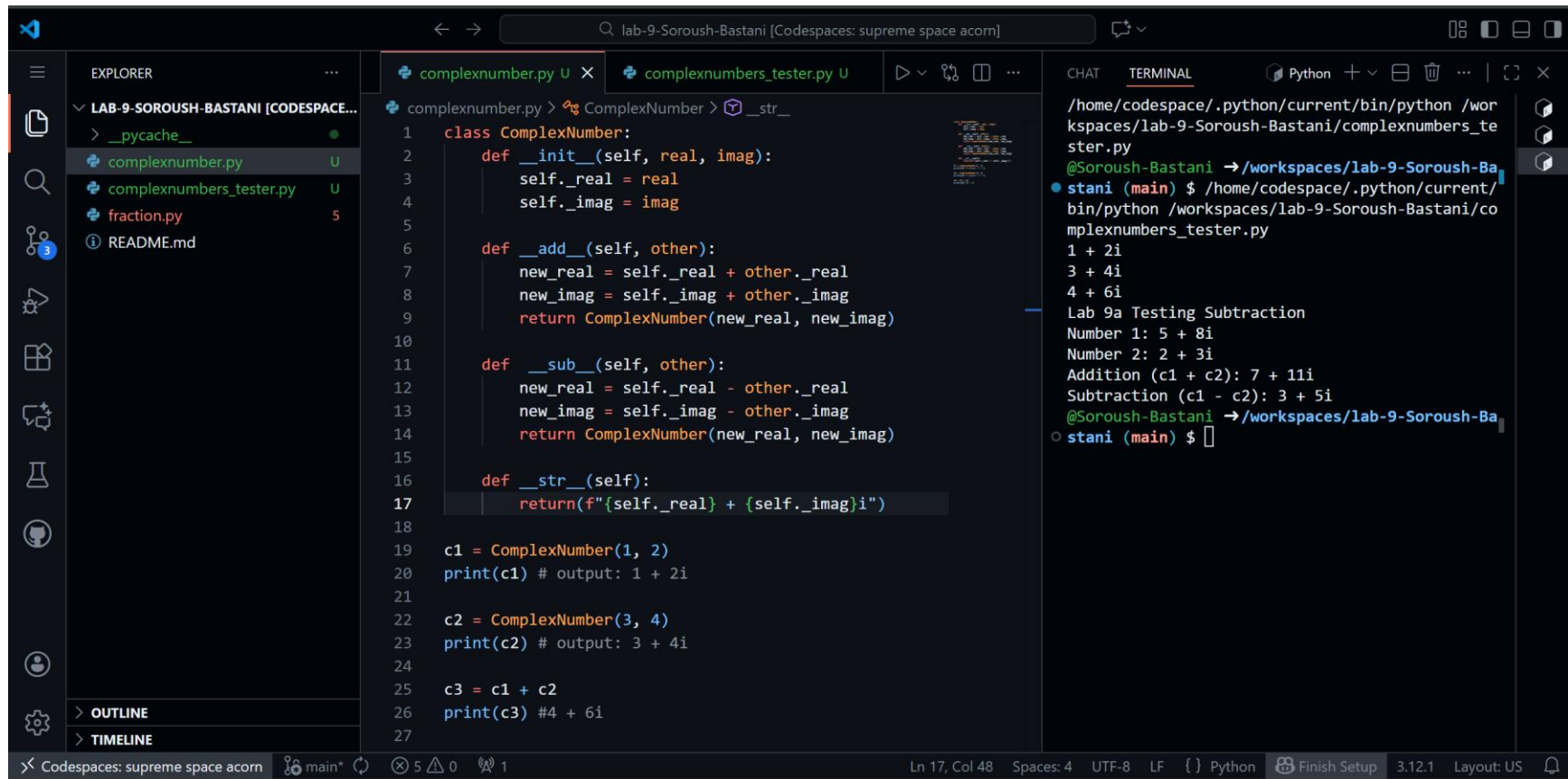


Lab 9a



The screenshot displays a VS Code editor window with the following components:

- EXPLORER:** Shows the file structure for 'LAB-9-SOROUSH-BASTANI [CODESPACE...]' with files: `__pycache__`, `complexnumber.py`, `complexnumbers_tester.py`, `fraction.py`, and `README.md`.
- EDITOR:** Displays the code for `complexnumber.py`. The code defines a `ComplexNumber` class with methods for initialization, addition, subtraction, and string representation. It also includes test cases for creating and adding complex numbers.
- TERMINAL:** Shows the execution of the test script. The output displays the results of the operations: `1 + 2i`, `3 + 4i`, and `4 + 6i`.

```
1 class ComplexNumber:
2     def __init__(self, real, imag):
3         self._real = real
4         self._imag = imag
5
6     def __add__(self, other):
7         new_real = self._real + other._real
8         new_imag = self._imag + other._imag
9         return ComplexNumber(new_real, new_imag)
10
11    def __sub__(self, other):
12        new_real = self._real - other._real
13        new_imag = self._imag - other._imag
14        return ComplexNumber(new_real, new_imag)
15
16    def __str__(self):
17        return(f"{self._real} + {self._imag}i")
18
19    c1 = ComplexNumber(1, 2)
20    print(c1) # output: 1 + 2i
21
22    c2 = ComplexNumber(3, 4)
23    print(c2) # output: 3 + 4i
24
25    c3 = c1 + c2
26    print(c3) #4 + 6i
27
```

```
/home/codespace/.python/current/bin/python /workspaces/lab-9-Soroush-Bastani/complexnumbers_tester.py
@Soroush-Bastani → /workspaces/lab-9-Soroush-Bastani
stani (main) $ /home/codespace/.python/current/bin/python /workspaces/lab-9-Soroush-Bastani/complexnumbers_tester.py
1 + 2i
3 + 4i
4 + 6i
Lab 9a Testing Subtraction
Number 1: 5 + 8i
Number 2: 2 + 3i
Addition (c1 + c2): 7 + 11i
Subtraction (c1 - c2): 3 + 5i
@Soroush-Bastani → /workspaces/lab-9-Soroush-Bastani
stani (main) $
```

SOROSH BASTANI
SBASTANI1

The image shows a VS Code editor interface with a dark theme. The Explorer panel on the left shows a file tree for 'LAB-9-SOROSH-BASTANI [CODESPACE...]'. The main editor area displays the file 'complexnumbers_tester.py' with the following Python code:

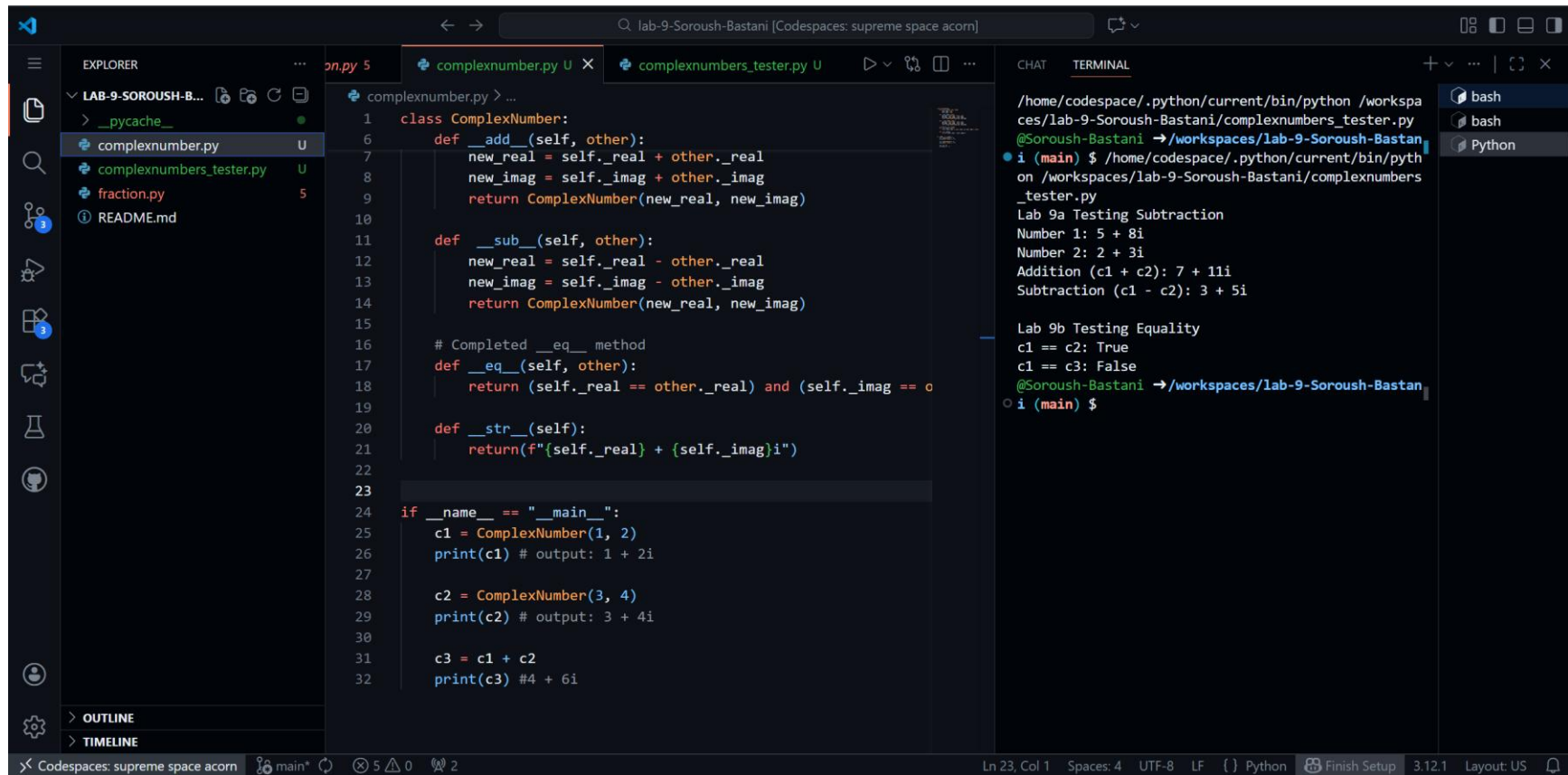
```
1 # Import the class we just made
2 from complexnumber import ComplexNumber
3
4 def main():
5     print("Lab 9a Testing Subtraction")
6
7     c1 = ComplexNumber(5, 8)
8     c2 = ComplexNumber(2, 3)
9
10    print(f"Number 1: {c1}")
11    print(f"Number 2: {c2}")
12
13    c3 = c1 + c2
14    print(f"Addition (c1 + c2): {c3}")
15
16    c4 = c1 - c2
17    print(f"Subtraction (c1 - c2): {c4}")
18
19 if __name__ == "__main__":
20     main()
```

The terminal on the right shows the command to run the script and its output:

```
/home/codespace/.python/current/bin/python /workspaces/lab-9-Sorosh-Bastani/complexnumbers_tester.py
@Sorosh-Bastani → /workspaces/lab-9-Sorosh-Bastani (main) $ /home/codespace/.python/current/bin/python /workspaces/lab-9-Sorosh-Bastani/complexnumbers_tester.py
1 + 2i
3 + 4i
4 + 6i
Lab 9a Testing Subtraction
Number 1: 5 + 8i
Number 2: 2 + 3i
Addition (c1 + c2): 7 + 11i
Subtraction (c1 - c2): 3 + 5i
@Sorosh-Bastani → /workspaces/lab-9-Sorosh-Bastani (main) $
```

The status bar at the bottom indicates the file is 'main.py', the editor is at 'Ln 8, Col 30', and the Python version is '3.12.1'.

Lab 9b



The screenshot displays a VS Code editor window with the following components:

- EXPLORER:** Shows the file structure of the project, including `__pycache__`, `complexnumber.py`, `complexnumbers_tester.py`, `fraction.py`, and `README.md`.
- EDITOR:** Displays the code for `complexnumber.py`. The code defines a `ComplexNumber` class with methods for addition, subtraction, equality checking, and string representation. It also includes a main block for testing the class.
- TERMINAL:** Shows the output of running the test script. It displays the results of addition and subtraction for two complex numbers, and the results of equality testing.

```
1 class ComplexNumber:
2     def __add__(self, other):
3         new_real = self._real + other._real
4         new_imag = self._imag + other._imag
5         return ComplexNumber(new_real, new_imag)
6
7     def __sub__(self, other):
8         new_real = self._real - other._real
9         new_imag = self._imag - other._imag
10        return ComplexNumber(new_real, new_imag)
11
12    # Completed __eq__ method
13    def __eq__(self, other):
14        return (self._real == other._real) and (self._imag == other._imag)
15
16    def __str__(self):
17        return f"{self._real} + {self._imag}i"
18
19
20 if __name__ == "__main__":
21     c1 = ComplexNumber(1, 2)
22     print(c1) # output: 1 + 2i
23
24     c2 = ComplexNumber(3, 4)
25     print(c2) # output: 3 + 4i
26
27     c3 = c1 + c2
28     print(c3) # output: 4 + 6i
```

Terminal Output:

```
/home/codespace/.python/current/bin/python /workspaces/lab-9-Soroush-Bastani/complexnumbers_tester.py
@Soroush-Bastani → /workspaces/lab-9-Soroush-Bastani
i (main) $ /home/codespace/.python/current/bin/python /workspaces/lab-9-Soroush-Bastani/complexnumbers_tester.py
Lab 9a Testing Subtraction
Number 1: 5 + 8i
Number 2: 2 + 3i
Addition (c1 + c2): 7 + 11i
Subtraction (c1 - c2): 3 + 5i

Lab 9b Testing Equality
c1 == c2: True
c1 == c3: False
@Soroush-Bastani → /workspaces/lab-9-Soroush-Bastani
i (main) $
```

SOROUSH BASTANI
SBASTANI1

The screenshot displays a VS Code editor window with the following components:

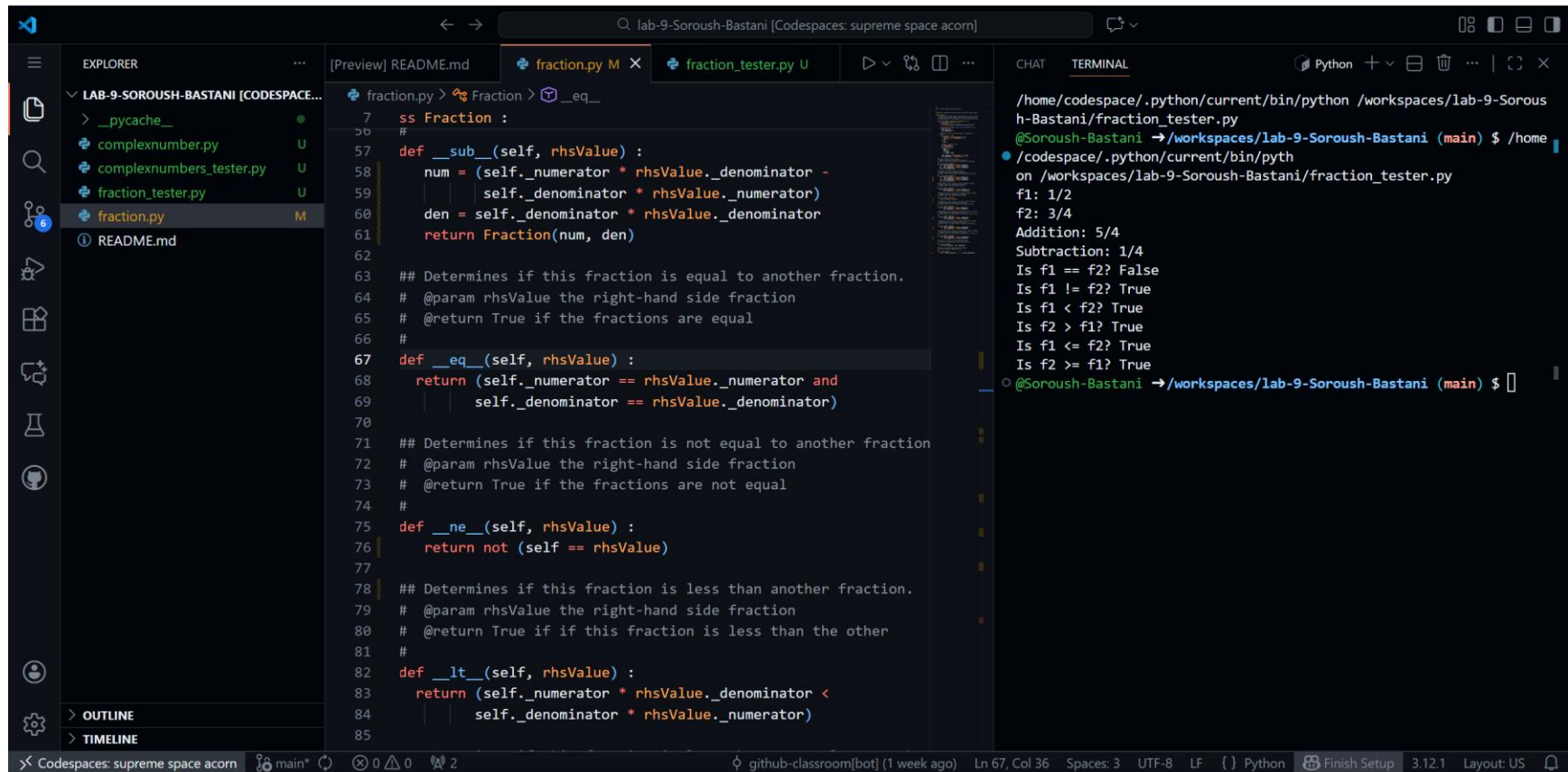
- EXPLORER:** Shows a file tree for 'LAB-9-SOROUSH-B...' containing files like `__pycache__`, `complexnumber.py`, `complexnumbers_tester.py`, `fraction.py`, and `README.md`.
- EDITOR:** Displays the `complexnumbers_tester.py` file with the following code:

```
1 from complexnumber import ComplexNumber
2
3
4 def main():
5     # --- Lab 9a: Testing Subtraction ---
6     print("Lab 9a Testing Subtraction")
7
8     c1 = ComplexNumber(5, 8)
9     c2 = ComplexNumber(2, 3)
10
11     print(f"Number 1: {c1}")
12     print(f"Number 2: {c2}")
13
14     c3 = c1 + c2
15     print(f"Addition (c1 + c2): {c3}")
16
17     c4 = c1 - c2
18     print(f"Subtraction (c1 - c2): {c4}")
19
20     # --- Lab 9b: Testing Equality ---
21     print("\nLab 9b Testing Equality")
22
23     c_eq_1 = ComplexNumber(3, 4)
24     c_eq_2 = ComplexNumber(3, 4)
25     c_eq_3 = ComplexNumber(2, 5)
26
27     print(f"c1 == c2: {c_eq_1 == c_eq_2}") # Expected: True
28     print(f"c1 == c3: {c_eq_1 == c_eq_3}") # Expected: False
29
30 if __name__ == "__main__":
31     main()
```
- TERMINAL:** Shows the execution output:

```
/home/codespace/.python/current/bin/python /workspa
ces/lab-9-Soroush-Bastani/complexnumbers_tester.py
@Soroush-Bastani → /workspaces/lab-9-Soroush-Bastan
i (main) $ /home/codespace/.python/current/bin/pyth
on /workspaces/lab-9-Soroush-Bastani/complexnumbe
rs_tester.py
Lab 9a Testing Subtraction
Number 1: 5 + 8i
Number 2: 2 + 3i
Addition (c1 + c2): 7 + 11i
Subtraction (c1 - c2): 3 + 5i

Lab 9b Testing Equality
c1 == c2: True
c1 == c3: False
@Soroush-Bastani → /workspaces/lab-9-Soroush-Bastan
i (main) $
```
- STATUS BAR:** Indicates the current file is `main*` at line 19, column 1, with 4 spaces, UTF-8 encoding, LF line endings, and Python syntax.

Lab 9c



The screenshot shows a VS Code editor window with the following components:

- EXPLORER:** Lists files in the `LAB-9-SOROUSH-BASTANI` workspace, including `fraction.py` (marked as modified).
- EDITOR:** Displays the `fraction.py` file. The code defines a `Fraction` class with methods for arithmetic and comparison.

```
7 class Fraction:
8     #
9     def __sub__(self, rhsValue):
10         num = (self._numerator * rhsValue._denominator -
11               self._denominator * rhsValue._numerator)
12         den = self._denominator * rhsValue._denominator
13         return Fraction(num, den)
14
15     ## Determines if this fraction is equal to another fraction.
16     # @param rhsValue the right-hand side fraction
17     # @return True if the fractions are equal
18     #
19     def __eq__(self, rhsValue):
20         return (self._numerator == rhsValue._numerator and
21               self._denominator == rhsValue._denominator)
22
23     ## Determines if this fraction is not equal to another fraction
24     # @param rhsValue the right-hand side fraction
25     # @return True if the fractions are not equal
26     #
27     def __ne__(self, rhsValue):
28         return not (self == rhsValue)
29
30     ## Determines if this fraction is less than another fraction.
31     # @param rhsValue the right-hand side fraction
32     # @return True if if this fraction is less than the other
33     #
34     def __lt__(self, rhsValue):
35         return (self._numerator * rhsValue._denominator <
36               self._denominator * rhsValue._numerator)
```
- TERMINAL:** Shows the execution of `fraction_tester.py` with the following output:

```
/home/codespace/.python/current/bin/python /workspaces/lab-9-Soroush-Bastani/fraction_tester.py
@Soroush-Bastani →/workspaces/lab-9-Soroush-Bastani (main) $ /home
/codespace/.python/current/bin/pyth
on /workspaces/lab-9-Soroush-Bastani/fraction_tester.py
f1: 1/2
f2: 3/4
Addition: 5/4
Subtraction: 1/4
Is f1 == f2? False
Is f1 != f2? True
Is f1 < f2? True
Is f2 > f1? True
Is f1 <= f2? True
Is f2 >= f1? True
@Soroush-Bastani →/workspaces/lab-9-Soroush-Bastani (main) $
```

SOROUSH BASTANI
SBASTANI1

The image shows a VS Code editor interface with a dark theme. The Explorer panel on the left shows a project named 'LAB-9-SOROUSH-B...' with files: `__pycache__`, `complexnumber.py`, `complexnumbers_tester.py`, `fraction_tester.py` (selected), `fraction.py`, and `README.md`. The main editor displays the content of `fraction_tester.py`, which is a Python script for testing fraction operations using the `Fraction` class from the `fractions` module. The script defines a `main()` function that creates two fractions, `f1 = Fraction(1, 2)` and `f2 = Fraction(3, 4)`, and performs various operations and comparisons. The `if __name__ == '__main__':` block calls the `main()` function. The output of the script is shown in the Terminal panel on the right, which displays the results of the operations and comparisons.

```
fraction_tester.py > ...
1  from fractions import Fraction
2
3  def main():
4      f1 = Fraction(1, 2)
5      f2 = Fraction(3, 4)
6
7      print("f1:", f1)
8      print("f2:", f2)
9
10     # Test Add and Sub
11     f3 = f1 + f2
12     print(f"Addition: {f3}")
13
14     f4 = f2 - f1
15     print(f"Subtraction: {f4}")
16
17     # Test Comparisons
18     print(f"Is f1 == f2? {f1 == f2}")
19     print(f"Is f1 != f2? {f1 != f2}")
20     print(f"Is f1 < f2? {f1 < f2}")
21     print(f"Is f2 > f1? {f2 > f1}")
22     print(f"Is f1 <= f2? {f1 <= f2}")
23     print(f"Is f2 >= f1? {f2 >= f1}")
24
25 if __name__ == '__main__':
26     main()
```

Terminal Output:

```
/home/codespace/.python/current/bin/python /workspaces/lab-9-Soroush-Bastani/fraction_tester.py
@Soroush-Bastani → /workspaces/lab-9-Soroush-Bastani (main) $ /home/codespace/.python/current/bin/python /workspaces/lab-9-Soroush-Bastani/fraction_tester.py
f1: 1/2
f2: 3/4
Addition: 5/4
Subtraction: 1/4
Is f1 == f2? False
Is f1 != f2? True
Is f1 < f2? True
Is f2 > f1? True
Is f1 <= f2? True
Is f2 >= f1? True
@Soroush-Bastani → /workspaces/lab-9-Soroush-Bastani (main) $
```

VS Code status bar at the bottom shows: Codespaces: supreme space acorn, main*, 0 errors, 0 warnings, 2 suggestions, Ln 26, Col 11, Spaces: 4, UTF-8, LF, Python, Finish Setup, 3.12.1, Layout: US.

Lab 9d

Working with the ComplexNumber and Fraction classes really clarified the difference between a class and an object for me. I realized that the class is essentially just a blueprint or a template, while the objects (like c1, c2, or f1) are the actual specific versions built from that blueprint. It was interesting to see how I could write the code once in the class, but create multiple objects that each held their own unique data without mixing them up.

The concept of operator overloading was definitely the coolest part of this lab. Before this, I thought + and - were fixed rules in Python. It feels powerful to know that I can define `__add__` to make the plus sign do whatever I want, like adding complex fractions. It made the testing code much easier to read because writing `f1 + f2` looks like a normal math equation, rather than having to call something clunky like `f1.add(f2)`.

Finally, I understand why encapsulation and `self._numerator` are so important. If we just used global variables, every object would share the same data, which would be a mess.

Using `self` ensures that the data stays safe and locked inside that specific object, so changing one fraction doesn't accidentally break another one.