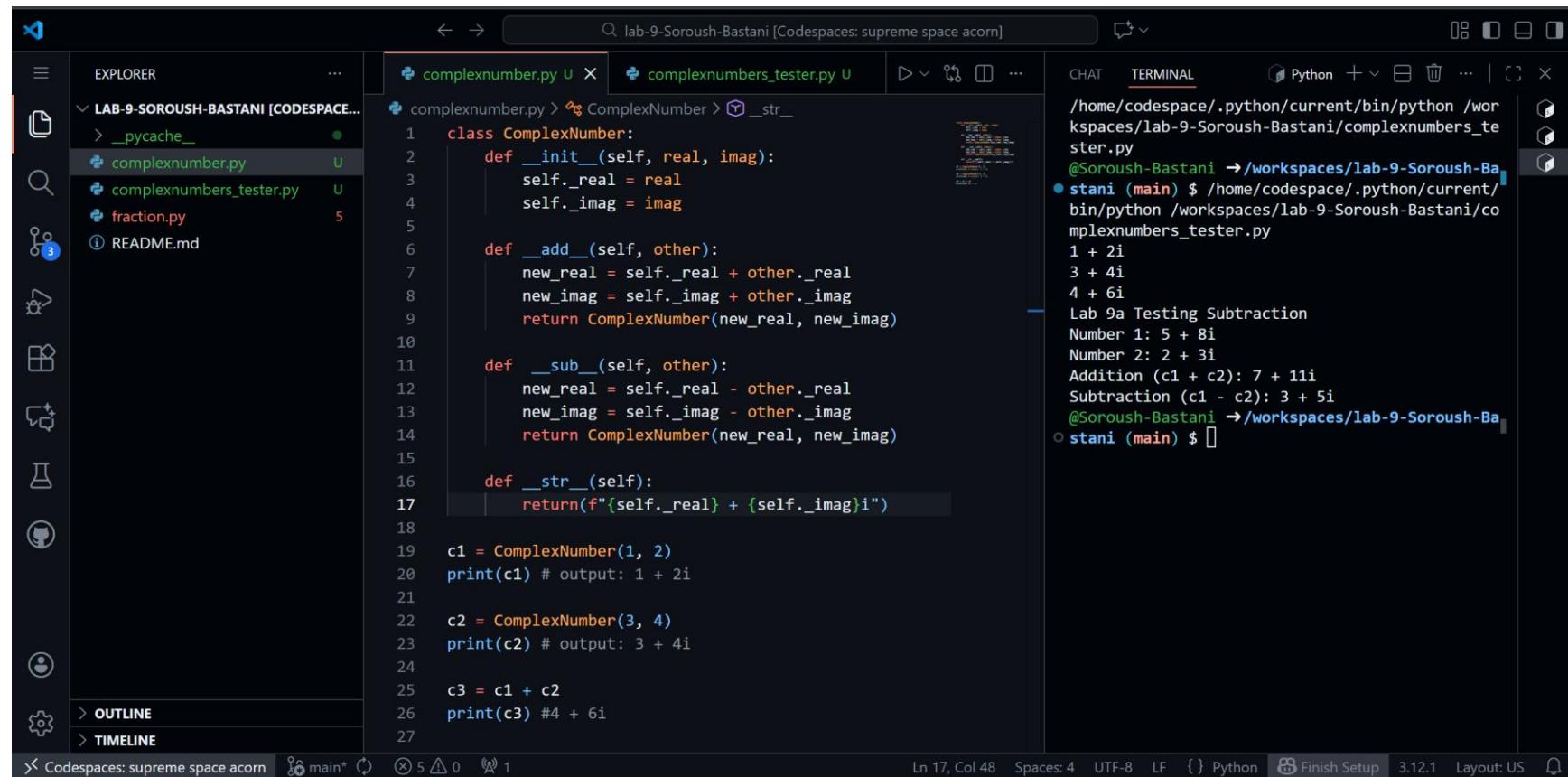


Lab 9a



The screenshot shows a terminal window within a code editor interface. The terminal is titled 'TERMINAL' and displays Python code for complex numbers and its execution.

```
/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) $ 
```

The code in 'complexnumber.py' defines a `ComplexNumber` class with methods for addition and subtraction, and a `__str__` method for string representation.

```
class ComplexNumber:
    def __init__(self, real, imag):
        self._real = real
        self._imag = imag

    def __add__(self, other):
        new_real = self._real + other._real
        new_imag = self._imag + other._imag
        return ComplexNumber(new_real, new_imag)

    def __sub__(self, other):
        new_real = self._real - other._real
        new_imag = self._imag - other._imag
        return ComplexNumber(new_real, new_imag)

    def __str__(self):
        return(f"{self._real} + {self._imag}i")
```

The code in 'complexnumbers_tester.py' creates instances of `ComplexNumber`, performs arithmetic operations, and prints the results.

```
c1 = ComplexNumber(1, 2)
print(c1) # output: 1 + 2i

c2 = ComplexNumber(3, 4)
print(c2) # output: 3 + 4i

c3 = c1 + c2
print(c3) #4 + 6i
```

SOROUSH BASTANI
SBASTANI1

The screenshot shows a Microsoft Visual Studio Code interface running in a Codespace. The title bar reads "lab-9-Soroush-Bastani [Codespaces: supreme space acorn]".

EXPLORER: Shows the project structure under "LAB-9-SOROUSH-BASTANI (CODESPACE...)". Files listed include: __pycache__, complexnumber.py (marked as updated), complexnumbers_tester.py (marked as updated), fraction.py, and README.md.

CODE EDITOR: The main editor area displays "complexnumbers_tester.py" with the following content:

```
complexnumbers_tester.py
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()
```

TERMINAL: The terminal tab is active, showing the output of running the script. It includes the command used to run the script, the imports, variable definitions, and the results of the addition and subtraction operations.

```
/home/codespace/.python/current/bin/python /workspaces/lab-9-Soroush-Bastani/complexnumbers_tester.py
@Soroush-Bastani → /workspaces/lab-9-Soroush-Bastani (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 (main) $
```

Lab 9b

The screenshot shows a terminal window with the following text:

```
/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
o i (main) $
```

The code in the terminal is a Python script named `complexnumbers_tester.py` which runs a series of tests on complex numbers. It includes imports for `ComplexNumber` and `ComplexNumbers`, and defines several methods for the `ComplexNumber` class, including `__add__`, `__sub__`, `__eq__`, and `__str__`. It also includes a main block with some test cases.

```
class ComplexNumber:
    def __add__(self, other):
        new_real = self._real + other._real
        new_imag = self._imag + other._imag
        return ComplexNumber(new_real, new_imag)

    def __sub__(self, other):
        new_real = self._real - other._real
        new_imag = self._imag - other._imag
        return ComplexNumber(new_real, new_imag)

    # Completed __eq__ method
    def __eq__(self, other):
        return (self._real == other._real) and (self._imag == other._imag)

    def __str__(self):
        return f"{self._real} + {self._imag}i"

if __name__ == "__main__":
    c1 = ComplexNumber(1, 2)
    print(c1) # output: 1 + 2i

    c2 = ComplexNumber(3, 4)
    print(c2) # output: 3 + 4i

    c3 = c1 + c2
    print(c3) #4 + 6i
```

SOROUSH BASTANI
SBASTANI1

The screenshot shows a terminal window with the following text:

```
/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) $
```

The terminal window is part of a larger interface, likely a code editor or terminal emulator, showing the file structure and code content.

EXPLORER pane:

- LAB-9-SOROUSH-B... (selected)
- pycache_
- complexnumber.py
- complexnumbers_tester.py (U)
- fraction.py
- README.md

CODE pane:

```
on.py 5 complexnumber.py U complexnumbers_tester.py U ...
```

```
from complexnumber import ComplexNumber

def main():
    # --- Lab 9a: Testing Subtraction ---
    print("Lab 9a Testing Subtraction")

    c1 = ComplexNumber(5, 8)
    c2 = ComplexNumber(2, 3)

    print(f"Number 1: {c1}")
    print(f"Number 2: {c2}")

    c3 = c1 + c2
    print(f"Addition (c1 + c2): {c3}")

    c4 = c1 - c2
    print(f"Subtraction (c1 - c2): {c4}")

    # --- Lab 9b: Testing Equality ---
    print("\nLab 9b Testing Equality")

    c_eq_1 = ComplexNumber(3, 4)
    c_eq_2 = ComplexNumber(3, 4)
    c_eq_3 = ComplexNumber(2, 5)

    print(f"c1 == c2: {c_eq_1 == c_eq_2}") # Expected: True
    print(f"c1 == c3: {c_eq_1 == c_eq_3}") # Expected: False

if __name__ == "__main__":
    main()
```

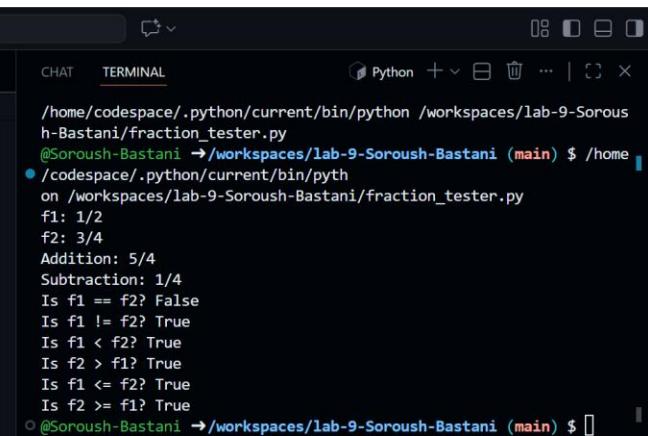
CHAT and **TERMINAL** panes are also visible at the top of the interface.

Lab 9c

The screenshot shows a GitHub Codespace interface with the following components:

- EXPLORER:** Shows files in the workspace, including `fraction.py` (modified), `fraction_tester.py` (untracked), `complexnumber.py`, `complexnumbers_tester.py`, and `README.md`.
- CODE EDITOR:** Displays the `fraction.py` file content. The code defines a `Fraction` class with methods for subtraction, equality comparison, inequality comparison, and less than comparison.
- TERMINAL:** Shows a Python session running fraction tests. It includes output for addition, subtraction, and comparison operations between fractions `f1` and `f2`.
- STATUS BAR:** Provides information about the workspace, including the file path (`/home/codespace/.python/current/bin/python /workspaces/lab-9-Soroush-Bastani/fraction_tester.py`), line and column numbers (Ln 67, Col 36), character encoding (UTF-8), and Python version (3.12.1).

SOROUSH BASTANI
SBASTANI1



```
from fraction import Fraction
def main():
    f1 = Fraction(1, 2)
    f2 = Fraction(3, 4)

    print("f1:", f1)
    print("f2:", f2)

    # Test Add and Sub
    f3 = f1 + f2
    print(f"Addition: {f3}")

    f4 = f2 - f1
    print(f"Subtraction: {f4}")

    # Test Comparisons
    print(f"Is f1 == f2? {f1 == f2}")
    print(f"Is f1 != f2? {f1 != f2}")
    print(f"Is f1 < f2? {f1 < f2}")
    print(f"Is f2 > f1? {f2 > f1}")
    print(f"Is f1 <= f2? {f1 <= f2}")
    print(f"Is f2 >= f1? {f2 >= f1}")

if __name__ == "__main__":
    main()
```

The screenshot shows a terminal window 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/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
```

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.