TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

ASSIGNMENT 4

By:

Sinjal Dahal (081/BEL/080)

DEPARTMENT OF COMPUTER ENGINEERING

LALITPUR, NEPAL

1. Addition Operator (+)

Documentation:

The addition operator (+) is overloaded for the ComplexNumber class to add two complex numbers or a complex number and a scalar. For two complex numbers, it adds the real and imaginary parts separately. For a scalar, it adds the scalar to the real part only, keeping the imaginary part unchanged.

Code:

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

    def __str__(self):
        return f"{self.real} + {self.imag}i"

    def __add__(self, other):
        if isinstance(other, (int, float)):
            return ComplexNumber(self.real + other, self.imag)
        return ComplexNumber(self.real + other.real, self.imag + other.imag)

if __name__ == "__main__":
    c1 = ComplexNumber(3, 2)
    c2 = ComplexNumber(1, 4)
    print("Addition:")
    print(f"c1 = {c1}")
    print(f"c2 = {c2}")
```

```
print(f"c1 + c2 = {c1 + c2}")
print(f"c1 + 5 = {c1 + 5}")
```

Output:

```
Addition:
c1 = 3 + 2i
c2 = 1 + 4i
c1 + c2 = 4 + 6i
c1 + 5 = 8 + 2i
```

2. Subtraction Operator (-)

Documentation:

The subtraction operator (-) subtracts one complex number from another or a scalar from a complex number. For two complex numbers, it subtracts the real and imaginary parts separately. For a scalar, it subtracts the scalar from the real part only, leaving the imaginary part unchanged.

Code:

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

    def __str__(self):
        return f"{self.real} + {self.imag}i"
```

```
    def __sub__(self, other):
        if isinstance(other, (int, float)):
            return ComplexNumber(self.real - other, self.imag)
        return ComplexNumber(self.real - other.real, self.imag - other.imag)

if __name__ == "__main__":
    c1 = ComplexNumber(3, 2)
    c2 = ComplexNumber(1, 4)
    print("Subtraction:")
    print(f"c1 = {c1}")
    print(f"c2 = {c2}")
    print(f"c1 - c2 = {c1 - c2}")
    print(f"c1 - 5 = {c1 - 5}")
```

Output:

```
Subtraction:
c1 = 3 + 2i
c2 = 1 + 4i
c1 - c2 = 2 + -2i
c1 - 5 = -2 + 2i
```

3. Multiplication Operator (*)

Documentation:

The multiplication operator (*) supports both complex number multiplication and scalar multiplication. For two complex numbers, it uses the formula (a + bi)(c + di) = (ac - bd) + (ad + bc)i. For a scalar, it multiplies both real and imaginary parts by the scalar.

Code:

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

    def __str__(self):
        return f"{self.real} + {self.imag}i"

    def __mul__(self, other):
        if isinstance(other, (int, float)):
            return ComplexNumber(self.real * other, self.imag * other)
        return ComplexNumber(
            self.real * other.real - self.imag * other.imag,
            self.real * other.imag + self.imag * other.real
        )

if __name__ == "__main__":
    c1 = ComplexNumber(3, 2)
    c2 = ComplexNumber(1, 4)
    print("Multiplication:")
    print(f"c1 = {c1}")
    print(f"c2 = {c2}")
```

```
print(f"c1 * c2 = {c1 * c2}")
print(f"c1 * 2 = {c1 * 2}")
```

Output:

```
Multiplication:
c1 = 3 + 2i
c2 = 1 + 4i
c1 * c2 = -5 + 14i
c1 * 2 = 6 + 4i
```

4. Division Operator (/)

Documentation

The division operator (/) supports division of complex numbers and division by a scalar. For complex numbers, it uses the conjugate method: $(a + bi)/(c + di) = ((a + bi)(c - di))/(c^2 + d^2)$. For a scalar, it divides both real and imaginary parts by the scalar.

Code:

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

    def __str__(self):
        return f"{self.real} + {self.imag}i"

    def __truediv__(self, other):
        if isinstance(other, (int, float)):
```

```
            return ComplexNumber(self.real / other, self.imag / other)
        denominator = other.real**2 + other.imag**2
        return ComplexNumber(
            (self.real * other.real + self.imag * other.imag) / denominator,
            (self.imag * other.real - self.real * other.imag) / denominator
        )

if __name__ == "__main__":
    c1 = ComplexNumber(3, 2)
    c2 = ComplexNumber(1, 4)
    print("Division:")
    print(f"c1 = {c1}")
    print(f"c2 = {c2}")
    print(f"c1 / c2 = {c1 / c2}")
    print(f"c1 / 2 = {c1 / 2}")
```

Output:

```
Division:
c1 = 3 + 2i
c2 = 1 + 4i
c1 / c2 = 0.29411764705882354 + -0.35294117647058826i
c1 / 2 = 1.5 + 1.0i
```

5. Floor Division Operator (//)

Documentation:

The floor division operator (//) performs integer division on the real and imaginary parts. For a scalar, it applies floor division to both parts. For complex numbers, it computes the division using the conjugate method and applies floor division to the result.

Code:

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

    def __str__(self):
        return f"{self.real} + {self.imag}i"

    def __floordiv__(self, other):
        if isinstance(other, (int, float)):
            return ComplexNumber(self.real // other, self.imag // other)
        denominator = other.real**2 + other.imag**2
        return ComplexNumber(
            (self.real * other.real + self.imag * other.imag) // denominator,
            (self.imag * other.real - self.real * other.imag) // denominator
        )

if __name__ == "__main__":
    c1 = ComplexNumber(3, 2)
    c2 = ComplexNumber(1, 4)
    print("Floor Division:")
```

```
print(f"c1 = {c1}")
print(f"c2 = {c2}")
print(f"c1 // c2 = {c1 // c2}")
print(f"c1 // 2 = {c1 // 2}")
```

Output:

```
Floor Division:
c1 = 3 + 2i
c2 = 1 + 4i
c1 // c2 = 0 + 0i
c1 // 2 = 1 + 1i
```

6. Modulo Operator (%)

Documentation:

The modulo operator (%) computes the remainder of division for real and imaginary parts. For a scalar, it applies modulo to both parts. For complex numbers, it computes the modulo of real and imaginary parts separately.

Code:

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

    def __str__(self):
        return f"{self.real} + {self.imag}i"
```

```
    def __mod__(self, other):
        if isinstance(other, (int, float)):
            return ComplexNumber(self.real % other, self.imag % other)
        return ComplexNumber(self.real % other.real, self.imag % other.imag)


if __name__ == "__main__":
    c1 = ComplexNumber(3, 2)
    c2 = ComplexNumber(1, 4)
    print("Modulo:")
    print(f"c1 = {c1}")
    print(f"c2 = {c2}")
    print(f"c1 % c2 = {c1 % c2}")
    print(f"c1 % 2 = {c1 % 2}")
```

Output:

```
Modulo:
c1 = 3 + 2i
c2 = 1 + 4i
c1 % c2 = 0 + 2i
c1 % 2 = 1 + 0i
```

7. Power Operator (**)

Documentation:

The power operator (**) raises the real and imaginary parts to a given power. For a scalar exponent, it raises each part to that power. For a complex number exponent, it raises the real and imaginary parts to the corresponding parts of the exponent.

Code:

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

    def __str__(self):
        return f"{self.real} + {self.imag}i"

    def __pow__(self, other):
        if isinstance(other, (int, float)):
            return ComplexNumber(self.real ** other, self.imag ** other)
        return ComplexNumber(self.real ** other.real, self.imag ** other.imag)

if __name__ == "__main__":
    c1 = ComplexNumber(3, 2)
    c2 = ComplexNumber(1, 4)
    print("Power:")
    print(f"c1 = {c1}")
    print(f"c2 = {c2}")
    print(f"c1 ** c2 = {c1 ** c2}")
    print(f"c1 ** 2 = {c1 ** 2}")
```

Output:

```
Power:
c1 = 3 + 2i
c2 = 1 + 4i
c1 ** c2 = 3 + 16i
c1 ** 2 = 9 + 4i
```

8. Equal Operator (==)

Documentation:

The equal operator (==) checks if two complex numbers have identical real and imaginary parts. It returns True only if both parts match exactly.

Code:

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

    def __str__(self):
        return f"{self.real} + {self.imag}i"

    def __eq__(self, other):
        return self.real == other.real and self.imag == other.imag

if __name__ == "__main__":
```

```
c1 = ComplexNumber(3, 2)
c2 = ComplexNumber(3, 2)
c3 = ComplexNumber(1, 4)
print("Equal:")
print(f"c1 = {c1}")
print(f"c2 = {c2}")
print(f"c3 = {c3}")
print(f"c1 == c2: {c1 == c2}")
print(f"c1 == c3: {c1 == c3}")
```

Output:

```
Equal:
c1 = 3 + 2i
c2 = 3 + 2i
c3 = 1 + 4i
c1 == c2: True
c1 == c3: False
```

9. Not Equal Operator (!=)

Documentation:

The not equal operator (!=) checks if two complex numbers differ in either their real or imaginary parts. It returns True if any part is different.

Code:

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

```
        self.imag = imag

    def __str__(self):
        return f"{self.real} + {self.imag}i"

    def __ne__(self, other):
        return not self.__eq__(other)

if __name__ == "__main__":
    c1 = ComplexNumber(3, 2)
    c2 = ComplexNumber(3, 2)
    c3 = ComplexNumber(1, 4)
    print("Not Equal:")
    print(f"c1 = {c1}")
    print(f"c2 = {c2}")
    print(f"c3 = {c3}")
    print(f"c1 != c2: {c1 != c2}")
    print(f"c1 != c3: {c1 != c3}")
```

Output:

```
Not Equal:
c1 = 3 + 2i
c2 = 3 + 2i
c3 = 1 + 4i
c1 != c2: False
c1 != c3: True
```

10. Less Than Operator (<)

Documentation:

The less than operator (<) compares the magnitudes of two complex numbers, where magnitude is sqrt(real² + imag²). It returns True if the first number's magnitude is less than the second's.

Code:

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

    def __str__(self):
        return f"{self.real} + {self.imag}i"

    def magnitude(self):
        return (self.real**2 + self.imag**2) ** 0.5

    def __lt__(self, other):
        return self.magnitude() < other.magnitude()

if __name__ == "__main__":
    c1 = ComplexNumber(3, 2)
    c2 = ComplexNumber(1, 4)
    print("Less Than:")
    print(f"c1 = {c1}")
    print(f"c2 = {c2}")
```

```
print(f"c1 < c2: {c1 < c2}")
```

Output:

```
Less Than:
c1 = 3 + 2i
c2 = 1 + 4i
c1 < c2: True
```

11. Greater Than Operator (>)

Documentation:

The greater than operator (>) compares the magnitudes of two complex numbers. It returns True if the first number's magnitude is greater than the second's.

Code:

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

    def __str__(self):
        return f"{self.real} + {self.imag}i"

    def magnitude(self):
        return (self.real**2 + self.imag**2) ** 0.5

    def __gt__(self, other):
```

```
        return self.magnitude() > other.magnitude()


if __name__ == "__main__":
    c1 = ComplexNumber(3, 2)
    c2 = ComplexNumber(1, 4)
    print("Greater Than:")
    print(f"c1 = {c1}")
    print(f"c2 = {c2}")
    print(f"c1 > c2: {c1 > c2}")
```

Output:

```
Greater Than:
c1 = 3 + 2i
c2 = 1 + 4i
c1 > c2: False
```

12. Less Than or Equal Operator (<=)

Documentation:

The less than or equal operator (<=) compares the magnitudes of two complex numbers. It returns True if the first number's magnitude is less than or equal to the second's.

Code:

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

```
        self.imag = imag

    def __str__(self):
        return f"{self.real} + {self.imag}i"

    def magnitude(self):
        return (self.real**2 + self.imag**2) ** 0.5

    def __le__(self, other):
        return self.magnitude() <= other.magnitude()

if __name__ == "__main__":
    c1 = ComplexNumber(3, 2)
    c2 = ComplexNumber(1, 4)
    print("Less Than or Equal:")
    print(f"c1 = {c1}")
    print(f"c2 = {c2}")
    print(f"c1 <= c2: {c1 <= c2}")
```

Output:

```
Less Than or Equal:
c1 = 3 + 2i
c2 = 1 + 4i
c1 <= c2: True
```

13. Greater Than or Equal Operator (>=)

Documentation:

The greater than or equal operator (>=) compares the magnitudes of two complex numbers. It returns True if the first number's magnitude is greater than or equal to the second's.

Code:

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

    def __str__(self):
        return f"{self.real} + {self.imag}i"

    def magnitude(self):
        return (self.real**2 + self.imag**2) ** 0.5

    def __ge__(self, other):
        return self.magnitude() >= other.magnitude()

if __name__ == "__main__":
    c1 = ComplexNumber(3, 2)
    c2 = ComplexNumber(1, 4)
    print("Greater Than or Equal:")
    print(f"c1 = {c1}")
    print(f"c2 = {c2}")
    print(f"c1 >= c2: {c1 >= c2}")
```

Output:

```
Greater Than or Equal:
c1 = 3 + 2i
c2 = 1 + 4i
c1 >= c2: False
```