Operaciones Básicas de Números Complejos en Python

> Por: Santiago Naranjo Herrera, David Santiago Sierra y Edgar Duván Bernal Acero

Debido a la longitud del código, se decidió no incluirlo en este documento. Se puede ver el código base en el siguiente Link: Código en Python

Introducción

En el análisis numérico y en muchas áreas de la ciencia y la ingeniería, los números complejos son fundamentales para resolver ecuaciones y representar fenómenos que no pueden ser descritos únicamente con números reales. Este informe se centra en la creación de un código en Python para realizar operaciones básicas (suma, resta, multiplicación y división) con números complejos. Se presentará un análisis del problema, la implementación del código, y se discutirá la precisión y eficiencia del mismo en el contexto del análisis numérico.

Análisis del Problema

Los números complejos se definen como una extensión de los números reales y se representan en la forma $a + bi$, donde a y b son números reales, y i es la unidad imaginaria, tal que $i^2 = -1$ Las operaciones básicas entre números complejos se definen de la siguiente manera:

- **Suma**: $(a + bi) + (c + di) = (a + c) + (b + d)i$
- **Resta:** $(a + bi) - (c + di) = (a - c) + (b - d)i$
- **Multiplicación:** $(a + bi) * (c + di) = (ac - bd) + (ad + dc)i$
- **División:** $\frac{a+bi}{c+di} = \frac{(a+bi)(c-di)}{c^2+d^2} = \frac{(ac+bd)+(bc-ad)i}{c^2+d^2}$

El principal desafío radica en implementar estas operaciones de manera precisa y eficiente, teniendo en cuenta las limitaciones de representación numérica en computadoras, como errores de redondeo y precisión limitada de los números en coma flotante, además del hecho de que no se pueden utilizar librerías externas del lenguaje.

Descripción de la Solución

El código implementa una clase llamada `NumeroComplejo` que define un número complejo.

```
(base) ➜ python /bin/python /home/santorar/Code/python/numerosComplejos.py
Ingrese el primer número complejo
Ingrese la parte real: 13
Ingrese la parte imaginaria: 45
Ingrese el segundo número complejo
Ingrese la parte real: 45
Ingrese la parte imaginaria: 13
```

Métodos para realizar las operaciones mencionadas:

- **Método** `__init__`: Inicializa la parte real e imaginaria del número complejo.
- **Método** `__str__`: Retorna una representación en cadena del número complejo en el formato $a + bi$.
- **Método** `sumar`: Suma dos números complejos.

```python
    def sumar(self, otro):
        """
        Suma este número complejo con otro.
        :param otro: Otro objeto NumeroComplejo.
        :return: Nuevo objeto NumeroComplejo con el resultado de la suma.
        """
        real = self.real + otro.real
        imag = self.imag + otro.imag
        return NumeroComplejo(real, imag)
```

```
Menú de operaciones con números complejos:
1. Sumar
2. Restar
3. Multiplicar
4. Dividir
5. Salir
Seleccione una opción (1-5): 1
Resultado de la suma: 58.0 + 58.0i
```

- **Método** `restar`: Resta un número complejo de otro.

```
    def restar(self, otro):
        """
        Resta otro número complejo a este.
        :param otro: Otro objeto NumeroComplejo.
        :return: Nuevo objeto NumeroComplejo con el resultado de la resta.
        """
        real = self.real - otro.real
        imag = self.imag - otro.imag
        return NumeroComplejo(real, imag)
```

```
Menú de operaciones con números complejos:
1. Sumar
2. Restar
3. Multiplicar
4. Dividir
5. Salir
Seleccione una opción (1-5): 2
Resultado de la resta: -32.0 + 32.0i
```

- **Método** `multiplicar`: Multiplica dos números complejos utilizando la fórmula de la multiplicación de números complejos.

```
    def multiplicar(self, otro):
        """
        Multiplica este número complejo por otro.
        :param otro: Otro objeto NumeroComplejo.
        :return: Nuevo objeto NumeroComplejo con el resultado de la
multiplicación.
        """
        real = self.real * otro.real - self.imag * otro.imag
        imag = self.real * otro.imag + self.imag * otro.real
        return NumeroComplejo(real, imag)
```

```
Menú de operaciones con números complejos:
1. Sumar
2. Restar
3. Multiplicar
4. Dividir
5. Salir
Seleccione una opción (1-5): 3
Resultado de la multiplicación: 0.0 + 2194.0i
```

- **Método** `dividir`: Divide dos números complejos, manejando el caso especial de la división por cero.

```
    def dividir(self, otro):
        """
        Divide este número complejo por otro.
        :param otro: Otro objeto NumeroComplejo.
        :return: Nuevo objeto NumeroComplejo con el resultado de la
división.
        """
        denominador = otro.real**2 + otro.imag**2
        real = (self.real * otro.real + self.imag * otro.imag) / denominador
        imag = (self.imag * otro.real - self.real * otro.imag) / denominador
        return NumeroComplejo(real, imag)
```

```
Menú de operaciones con números complejos:
1. Sumar
2. Restar
3. Multiplicar
4. Dividir
5. Salir
Seleccione una opción (1-5): 4
Resultado de la división: 0.5332725615314494 + 0.8459434822248i
```

Además, el código contiene funciones auxiliares para la interacción con el usuario:

- **Función** `mostrar_menu`: Presenta un menú para que el usuario seleccione una operación.
- **Función** `solicitar_numero_complejo`: Solicita al usuario que ingrese un número complejo.

```
def solicitar_numero_complejo():
    """
    Solicita al usuario ingresar un número complejo.
    :return: Objeto NumeroComplejo con los valores ingresados por el
usuario.
    """
    real = float(input("Ingrese la parte real: "))
    imag = float(input("Ingrese la parte imaginaria: "))
    return NumeroComplejo(real, imag)
```

- **Función** `main`: Controla el flujo del programa, permitiendo al usuario elegir una operación y mostrando el resultado correspondiente.

```
def mostrar_menu():
    """
    Muestra el menú de opciones al usuario.
    """
    print("\nMenú de operaciones con números complejos:")
    print("1. Sumar")
    print("2. Restar")
    print("3. Multiplicar")
    print("4. Dividir")
    print("5. Salir")
```

Análisis Numérico

Desde la perspectiva del análisis numérico, es importante destacar varios puntos relevantes:

Precisión de las Operaciones

Todas las operaciones aritméticas con números complejos dependen de la precisión del tipo `float` en Python. Sin embargo, la suma, resta y multiplicación no presentan problemas de precisión significativos en la mayoría de los casos. La división, por otro lado, puede tener problemas de precisión si el denominador es muy pequeño, debido al fenómeno de la pérdida de significancia.

Complejidad Computacional

La complejidad de las operaciones es constante O(1)O(1)O(1) ya que solo se realizan un número fijo de multiplicaciones y sumas para cada operación. Esto hace que las operaciones sean eficientes incluso para un gran número de cálculos consecutivos.

Estabilidad Numérica

La división de números complejos puede ser inestable cuando el denominador es cercano a cero. Esto se mitiga en parte verificando explícitamente si el denominador es cero y evitando la división en tales casos. Sin embargo, para denominadores muy pequeños pero no exactamente cero, puede haber inestabilidad numérica. En tales casos, podría ser conveniente utilizar una biblioteca especializada en números complejos que maneje estos casos con mayor precisión.

Conclusión

La implementación del manejo de números complejos en Python es adecuada para aplicaciones generales y educativas. Proporciona una base sólida para entender las operaciones con números complejos y cómo se implementan en software. Desde el punto de vista del análisis numérico, esta solución es eficiente y precisa en la mayoría de los casos prácticos. Sin embargo, en aplicaciones donde se requiere alta precisión, especialmente cuando se manipulan números complejos con denominadores muy pequeños, puede ser preferible utilizar bibliotecas especializadas como `numpy` o `cmath` para aprovechar algoritmos más robustos y optimizados.

En general, este código cumple con los requisitos de un uso básico y es una herramienta útil para comprender las operaciones fundamentales con números complejos en programación y análisis matemático.