Operadores

Programação Orientada a Objetos

Python

Prof. Diógenes Furlan

Sumário

- Operadores Aritméticos
- Operadores Relacionais
- Operadores de Sequencia
- Conversores de Tipo

Métodos Mágicos

- Operadores podem ser sobrecarregados em classes através de métodos especiais conhecidos como **métodos mágicos** ou **métodos especiais**.
  - iniciam e terminam por __(2 underlines)

Operadores Aritméticos

- Adição: `__add__(self, other)`
- Subtração: `__sub__(self, other)`
- Multiplicação: `__mul__(self, other)`
- Divisão: `__truediv__(self, other)`
- Divisão Inteira: `__floordiv__(self, other)`
- Resto da Divisão: `__mod__(self, other)`
- Potência: `__pow__(self, other)`
- E lógico: `__and__(self, other)`
- OU lógico: `__or__(self, other)`

Operadores Aritméticos Unários

- Adição: `__pos__(self)`
  - Este método é chamado quando um objeto é precedido por um sinal de adição (+).
- Subtração: `__neg__(self)`
  - Este método é chamado quando um objeto é precedido por um sinal de subtração (-).
- Absoluto: `__abs__(self)`
- Negação lógica: `__not__(self)`

Operadores Relacionais

- Igualdade: `__eq__(self, other)`
- Diferença: `__ne__(self, other)`
- Menor que: `__lt__(self, other)`
- Menor ou igual a: `__le__(self, other)`
- Maior que: `__gt__(self, other)`
- Maior ou igual a: `__ge__(self, other)`

Operadores de Sequencia

- Acesso por Índice: `__getitem__(self, key)`
- Definição por Índice: `__setitem__(self,key,value)`
- Deleção por Índice: `__delitem__(self, key)`
- Comprimento (len): `__len__(self)`
- Iteração (iter): `__iter__(self)`

Conversores

- Conversão para String: `__str__(self)`
- "Representação oficial" em string do objeto: `__repr__(self)`
- Conversão para Inteiro: `__int__(self)`
- Conversão para Float: `__float__(self)`
- Conversão para Booleano: `__bool__(self)`

Exemplo 1

```
Pos =  5
Neg =  -5
Abs =  5
Pos =  -10
Neg =  10
Abs =  10
```

```python
class Numero:
    def __init__(self, valor):
        self.valor = valor

    def __pos__(self):
        return +self.valor

    def __neg__(self):
        return -self.valor

    def __abs__(self):
        return abs(self.valor)
```

```python
def main():
    num1 = Numero(5)
    num2 = Numero(-10)

    print("Pos = ", +num1)
    print("Neg = ", -num1)
    print("Abs = ", abs(num1))
    print("Pos = ", +num2)
    print("Neg = ", -num2)
    print("Abs = ", abs(num2))

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

Exemplo 2 – Operadores de Sequencia

- Faça uma classe Playlist, que contenha uma lista de nomes de músicas
- Para que a classe funcione "como uma lista", redefina os métodos:

```
__detitem__
__setitem__
__delitem__
__len__
```

Exemplo 2 – Operadores de Sequencia

```
1  class Playlist:
2      def __init__(self):
3          self.musics = []
4
5      def __getitem__(self, index):
6          return self.musics[index]
7
8      def __setitem__(self, index, music):
9          self.musics[index] = music
10
11     def __len__(self):
12         return len(self.musics)
13
14     def adicionar_musica(self, music):
15         self.musics.append(music)
```

Exemplo 2 – Operadores de Sequencia

```
18 # main
19 playlist = Playlist()
20 playlist.adicionar_musica("Thriller")
21 playlist.adicionar_musica("Planeta água")
22 playlist.adicionar_musica("Amadeo")
23
24 print("Músicas na playlist:")
25 for musica in playlist:
26     print("-", musica)
27
28 print("Músicas na playlist:")
29 for i in range( len(playlist) ):
30     print("-", playlist[i])
```

Exemplo 3 – Classe Fração

- Faça uma classe Fração com os seguintes operadores:
  - Adição (de 2 fracionários)
  - Subtração (de 2 fracionários)
  - Multiplicação (de 2 fracionários)
  - Divisão (de 2 fracionários)
  - Comparação de igualdade
  - Comparação de maior que
  - Comparação de menor que
  - Conversão para real
  - Conversão para int
  - Conversão para string