

## Módulo 07

### Métodos Mágicos (Aula 9)



#### Questões de Aprendizagem

##### Questão 1:

##### **Criando uma classe que suporta comparações (< <= > >= == !=)**

Implemente uma classe chamada Produto que representa um produto com nome e preço. A classe deve permitir comparações entre dois objetos Produto com base no preço.

Implemente os seguintes métodos mágicos:

- `__lt__` (menor que)
- `__le__` (menor ou igual)
- `__eq__` (igual)
- `__ne__` (diferente)
- `__gt__` (maior que)
- `__ge__` (maior ou igual)

##### **Exemplo de uso:**

```
p1 = Produto("Notebook", 3000) # Preço: 3000
p2 = Produto("Smartphone", 1500) # Preço: 1500

print(f"p1 < p2: {p1 < p2}") # Saída esperada: False
print(f"p1 > p2: {p1 > p2}") # Saída esperada: True
print(f"p1 == p2: {p1 == p2}") # Saída esperada: False
```

## Módulo 07

### Métodos Mágicos (Aula 9)



#### Questões de Aprendizagem

##### Questão 2:

##### **Criando uma classe que suporta cálculos numéricos (+ - \* /)**

Implemente uma classe chamada Saldo que representa o saldo de uma conta bancária. A classe deve permitir:

1. Somar dois objetos Saldo (`__add__`).
2. Subtrair um objeto Saldo de outro (`__sub__`).

##### **Exemplo de uso:**

```
s1 = Saldo(500) # Saldo de 500
s2 = Saldo(200) # Saldo de 200

s3 = s1 + s2
print(f"Tipo de s3 é: {type(s3)}") # Saída esperada: <class
'__main__.Saldo'> (o __main__ pode variar)
print(f"Soma dos saldos: {s3.valor}") # Saída esperada: 700

s4 = s1 - s2
print(f"Tipo de s4 é: {type(s4)}") # Saída esperada: <class
'__main__.Saldo'> (o __main__ pode variar)
print(f"Subtração dos saldos: {s4.valor}") # Saída esperada: 300
```

## Módulo 07

### Métodos Mágicos (Aula 9)



#### Questões de Aprendizagem

##### Questão 3:

##### Criando uma classe que suporta o comando "len()"

Implemente uma classe chamada Livro que representa um livro com capítulos. A classe deve:

1. Armazenar capítulos em uma lista interna privada.
2. Retornar o número de capítulos ao usar a função len() na instância da classe (\_\_len\_\_).

##### Exemplo de uso:

```
livro = Livro()
livro.adicionar_capitulo("Capítulo 1: Introdução")
livro.adicionar_capitulo("Capítulo 2: Desenvolvimento")

print(f"Número de capítulos: {len(livro)}") # Saída esperada: 2
```

##### Questão 4:

##### Criando uma classe suporta o comando "in"

Implemente uma classe chamada Biblioteca que representa um acervo de livros. A classe deve permitir:

1. Adicionar livros ao acervo.
2. Verificar se um livro está na biblioteca usando a palavra-chave in (\_\_contains\_\_).

##### Exemplo de uso:

```
biblioteca = Biblioteca()
biblioteca.adicionar_livro("O Pequeno Príncipe")
biblioteca.adicionar_livro("1984")

print("O Pequeno Príncipe" in biblioteca) # Saída esperada: True
print("Dom Quixote" in biblioteca)       # Saída esperada: False
```

## Módulo 07

### Métodos Mágicos (Aula 9)



#### Questões de Aprendizagem

##### Questão 5:

##### **Criando uma classe que imita um dicionário**

Implemente uma classe chamada `Inventario` que simula um inventário de itens de uma loja. A classe deve permitir:

1. Adicionar itens usando o operador de atribuição com índice (`__setitem__`).
2. Recuperar a quantidade de um item usando o operador de acesso com índice (`__getitem__`). Se o produto não estiver no inventário, deve retornar 0.
3. Incrementar a quantidade de um item usando o operador de incremento (`__iadd__`).
4. Decrementar a quantidade de um item usando o operador de decremento (`__isub__`). Se a quantidade ficar negativa, deve ser ajustada para 0.

##### **Exemplo de uso:**

## Módulo 07

### Métodos Mágicos (Aula 9)



#### Questões de Aprendizagem

```
# Criar o inventário
inventario = Inventario()

# Adicionar itens
inventario['maçã'] = 50
inventario['banana'] = 100

# Recuperar a quantidade de um item
print(f"Quantidade de maçãs: {inventario['maçã']}") # Saída esperada: 50
print(f"Quantidade de bananas: {inventario['banana']}") # Saída esperada: 100

# Remover um item
del inventario['banana']

# Recuperar a quantidade de um item que não está no inventário
print(f"Quantidade de batatas: {inventario['batata']}") # Saída esperada: 0
print(f"Quantidade de bananas: {inventario['banana']}") # Saída esperada: 0

# Incrementar a quantidade de um item
inventario['maçã'] += 5

print(f"Quantidade de maçãs após incremento: {inventario['maçã']}") #
Saída esperada: 55

# Decrementar a quantidade de um item
inventario['maçã'] -= 10

print(f"Quantidade de maçãs após decremento: {inventario['maçã']}") #
Saída esperada: 45
```

## Módulo 07

### Métodos Mágicos (Aula 9)



#### Questões de Aprendizagem

##### Questão 6:

##### Criando uma classe que suporta o comando "with"

Implemente uma classe chamada `ArquivoTemporario` que cria e exclui um arquivo temporário automaticamente. A classe deve:

1. Criar o arquivo no método `__enter__`.
2. Excluir o arquivo no método `__exit__`.

##### Exemplo de uso:

```
with ArquivoTemporario("exemplo.txt") as arquivo:
    arquivo.write("Conteúdo temporário")
# O arquivo "exemplo.txt" é excluído automaticamente após o bloco `with`.
```

##### Questão 7:

##### Criando uma classe que suporta um loop for

Implemente uma classe chamada `Contador` que representa uma sequência de números de um valor inicial a um valor final (como se fosse um `range`). A classe deve implementar os métodos `__iter__` e `__next__` para permitir iteração.

##### Exemplo de uso:

```
contador = Contador(1, 5)

for numero in contador:
    print(numero)
# Saída esperada:
# 1
# 2
# 3
# 4
# 5
```

## Módulo 07

### Métodos Mágicos (Aula 9)



#### Questões de Aprendizagem

##### Questão 8:

Implemente uma classe chamada `Colecao` que combina vários métodos mágicos:

1. Armazena itens em uma lista interna privada.
2. Suporte acesso por índice (`__getitem__`), verificação de tamanho (`__len__`), e verificação de itens (`__contains__`).
3. Permita iterar sobre os itens com `__iter__` e `__next__`.

##### Exemplo de uso:

```
colecão = Colecao()
colecão.adicionar_item("Item 1")
colecão.adicionar_item("Item 2")

print(f"Tamanho da coleção: {len(colecão)}") # Saída esperada: 2
print(f"Item 1 está na coleção? {'Item 1' in coleção}") # Saída esperada:
True

for item in coleção:
    print(item)
# Saída esperada:
# Item 1
# Item 2
```



## Módulo 07

### Métodos Mágicos (Aula 9)



#### Questões de Aprendizagem

##### Questão 9:

Implemente uma classe chamada `Tempo` que representa um período em horas e minutos. A classe deve atender aos seguintes requisitos:

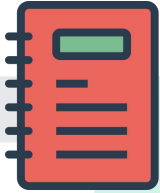
1. **Inicialização:**
  - Permitir a criação de objetos `Tempo` utilizando dois valores inteiros: um representando as horas e outro os minutos.
2. **Operações matemáticas:**
  - Implementar a soma (+) entre dois objetos `Tempo`, retornando um novo objeto `Tempo` que represente a soma das durações.
  - Implementar a subtração (-) entre dois objetos `Tempo`, retornando um novo objeto `Tempo` que represente a diferença das durações. Se o resultado for negativo, a operação deve lançar um erro do tipo `ValueError` com a mensagem "Tempo não pode ser negativo".
3. **Comparação:**
  - Implementar os operadores de comparação (<, >, <=, >=, ==, !=) entre dois objetos `Tempo`, com base no total de minutos.
4. **Representação em string:**
  - Implementar o método mágico `__str__` para representar o tempo no formato "HH:MM", onde HH e MM são preenchidos com zeros à esquerda, se necessário.

**Exemplo de uso esperado:**



## Módulo 07

### Métodos Mágicos (Aula 9)



#### Questões de Aprendizagem

```
# Criação de objetos Tempo
t1 = Tempo(2, 30) # Representa 2 horas e 30 minutos
t2 = Tempo(1, 45) # Representa 1 hora e 45 minutos

# Representação em string
print(f"t1 = {t1}") # Saída esperada: "t1 = 02:30"
print(f"t2 = {t2}") # Saída esperada: "t2 = 01:45"

# Soma de tempos
t3 = t1 + t2
print(f"t1 + t2 = {t3}") # Saída esperada: "t1 + t2 = 04:15"

# Subtração de tempos
t4 = t1 - t2
print(f"t1 - t2 = {t4}") # Saída esperada: "t1 - t2 = 00:45"

# Comparações
print(f"t1 < t2: {t1 < t2}") # Saída esperada: "t1 < t2: False"
print(f"t1 > t2: {t1 > t2}") # Saída esperada: "t1 > t2: True"

# Subtração resultando em erro
try:
    t5 = t2 - t1
except ValueError as e:
    print(e) # Saída esperada: "Tempo não pode ser negativo"
```

## Módulo 07

### Métodos Mágicos (Aula 9)



#### Questões de Aprendizagem

##### Questão 10:

Implemente uma classe chamada `Matriz` para representar uma matriz (pesquise o que é matriz caso não conheça). A classe deve atender aos seguintes requisitos:

1. **Inicialização:**
  - Permitir a criação de uma matriz de tamanho fixo, informando o número de linhas e colunas no momento da criação. Utilize uma lista de listas (privada) para representar a matriz.
  - Inicialize todos os elementos da matriz com o valor `0`.
2. **Acesso a elementos:**
  - Implementar os métodos mágicos `__getitem__` e `__setitem__` para acessar e modificar elementos da matriz usando índices no formato `(i, j)`.
  - Caso um índice esteja fora dos limites da matriz, a operação deve lançar um erro do tipo `IndexError` com a mensagem "Índice fora dos limites".
3. **Representação em string:**
  - Implementar o método mágico `__str__` para exibir a matriz completa em um formato tabular.

**Exemplo de uso esperado:**

## Módulo 07

### Métodos Mágicos (Aula 9)



#### Questões de Aprendizagem

```
# Criação de uma matriz 3x3
matriz = Matriz(3, 3)

# Inserção de valores
matriz[0, 0] = 1 # obs: "0, 0" é implicitamente uma tupla, é equivalente a matriz[(0, 0)]
matriz[1, 1] = 2
matriz[2, 2] = 3

# Representação da matriz
print("Matriz:")
print(matriz)
# Saída esperada:
# 1 0 0
# 0 2 0
# 0 0 3

# Acesso a elementos
print(f"Elemento na posição (1, 1): {matriz[1, 1]}") # Saída esperada: 2
print(f"Elemento na posição (0, 2): {matriz[0, 2]}") # Saída esperada: 0

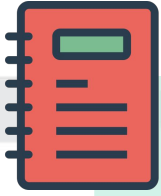
# Modificação de valores
matriz[0, 2] = 4
print("Matriz atualizada:")
print(matriz)
# Saída esperada:
# 1 0 4
# 0 2 0
# 0 0 3

# Alteração de valores para zero
matriz[0, 0] = 0
print("Matriz após alteração de um elemento:")
print(matriz)
# Saída esperada:
# 0 0 4
# 0 2 0
# 0 0 3

# ... continua
```

## Módulo 07

### Métodos Mágicos (Aula 9)



#### Questões de Aprendizagem

```
# Índices fora dos limites
try:
    matriz[3, 3] = 5
except IndexError as e:
    print(e) # Saída esperada: "Índice fora dos limites"
```