Listas em Objetos
Sobrecarga de Operadores

CIÊNCIA DA COMPUTAÇÃO

ATITUS EDUCAÇÃO

- Aula 05 -
Organização e Abstração

Prof. Me. Lucas R. C. Pessutto

ATITUS EDUCAÇÃO

Problema: Game Score

Passo 1: Classe ItemPlacar

* Implemente uma classe para representar um Item do placar.
* Cada item possui o nome do jogador e o placar obtido
* Faça o método `__str__` que retorna a linha do placar formatada
* Utilize encapsulamento!

Passo 1: Classe ItemPlacar

```python
class ItemPlacar:
    def __init__(self, nome, placar):
        self._nome = nome
        self._placar = placar

    @property
    def nome(self):
        return self._nome

    @nome.setter
    def nome(self, valor):
        self._nome = valor

    @property
    def placar(self):
        return self._placar

    @placar.setter
    def placar(self, valor):
        self._placar = valor
```

```python
def __str__(self):
    return f"{self._nome:10}\t{self._placar:5}"
```

Passo 2: Classe Placar

* Na classe Placar recebemos como parâmetro o tamanho máximo do placar (Utilize como valor padrão 5).

* No construtor dessa classe, crie uma lista vazia para armazenar os itens de score

* Defina o método `__str__` dessa classe, que retorna todo o placar

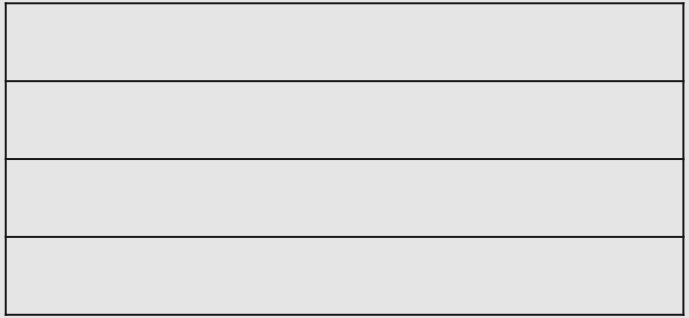
* Pense em como fazer a implementação do método para inserir um item no placar:

```
def inserir_item(self, item: ItemPlacar):
```

Método Inserir

```
placar = Placar(4)
placar.inserir_item(ItemPlacar("AAA", 123))
placar.inserir_item(ItemPlacar("BBB", 234))
placar.inserir_item(ItemPlacar("CCC", 200))
placar.inserir_item(ItemPlacar("DDD", 100))
placar.inserir_item(ItemPlacar("EEE", 400))
placar.inserir_item(ItemPlacar("FFF", 99))
placar.inserir_item(ItemPlacar("GGG", 150))
```

```
ItemPlacar("AAA", 123)
```



Método Inserir

* Seria mais fácil implementar esse método utilizando as funções de lista!

```
def inserir_item(self, item: ItemPlacar):
        self._scores.append(item)
        self._scores.sort(reverse=True)
        if len(self._scores) > self._max_items:
            self._scores = self._scores[:self._max_items]
```

* Porém, o método `sort` não sabe como comparar dois ItemPlacar!

Método Inserir

* O método sort utiliza os operadores de comparação (>, <, ==, !=, etc.) para comparar objetos

* Podemos dizer como esses operadores se comportam na nossa classe sobrescrevendo certas funções mágicas:

a < b	a.__lt__(b)
a <= b	a.__le__(b)
a > b	a.__gt__(b)
a >= b	a.__ge__(b)
a == b	a.__eq__(b)
a != b	a.__ne__(b)

Método Inserir

```
def __lt__(self, outroItem):
    return self._placar < outroItem._placar

def __gt__(self, outroItem):
    return self._placar > outroItem._placar

def __eq__(self, outroItem):
    return self._placar == outroItem._placar

def __le__(self, outroItem):
    return self._placar <= outroItem._placar

def __ge__(self, outroItem):
    return self._placar >= outroItem._placar

def __ne__(self, outroItem):
    return self._placar != outroItem._placar
```

```
p1 = ItemPlacar("AAA", 123)

p2 = ItemPlacar("ZZZ", 123)

print(p1 == p2)
print(p1 > p2)
print(p1 <= p2)
```

Sobrecarga de Operadores

* Podemos fazer nossa classe Placar se comportar como uma lista sobrescrevendo alguns métodos

v in a	a.__contains__(v)
a[k]	a.__getitem__(k)
a[k] = v	a.__setitem__(k,v)
del a[k]	a.__delitem__(k)
len(a)	a.__len__()

Outros Métodos

* Operadores Matemáticos:

a + b	a.__add__(b);	alternatively b.__radd__(a)
a − b	a.__sub__(b);	alternatively b.__rsub__(a)
a * b	a.__mul__(b);	alternatively b.__rmul__(a)
a / b	a.__truediv__(b);	alternatively b.__rtruediv__(a)
a // b	a.__floordiv__(b);	alternatively b.__rfloordiv__(a)
a % b	a.__mod__(b);	alternatively b.__rmod__(a)
a ** b	a.__pow__(b);	alternatively b.__rpow__(a)

Outros Métodos

* Conversões para tipos básicos do Python:

bool(a)	a.__bool__()
float(a)	a.__float__()
int(a)	a.__int__()

Sobrecarga de Operadores

* A sobrecarga de operadores costuma ser utilizada em três ocasiões especiais:
  - → Para emular tipos numéricos que não existem no Python
  - → Converter um objeto para um tipo básico do Python (str, int, etc.)
  - → Implementar coleções