OOP (Programação Orientado a Objetos)

Um detalhe importantíssimo que talvez não tenha percebido, mas embora utilizarmos o **Python** até agora como uma linguagem **funcional** e **procedural**, mas o **Python** na verdade é uma linguagem de **Programação Orientado a Objetos**!!

Isso porque no Python, tudo é exatamente **OBJETO**!! Porém como o **Python** se trata de uma linguem bem flexível, então nos permite trabalhar também como uma linguagem **procedural** e **funcional**, porém a sua essência é **orientado a objeto**!!

Criando a primeira classe

Veremos agora como criar uma classe simples em Python, e veja como é simples!

```
# Criando a primeira classe
class Funcionario:
    nome: str
    sobrenome: str

func1 = Funcionario()
func1.nome = 'Fernando'
func1.sobrenome = 'Spelling'

print(func1.nome)
```

```
Fernando
```

Veja que basicamente criamos uma classe com o nome **Funcionário** com 2 atributos (**nome** e **sobrenome**). E que após a criação do objeto, definimos os valores de seus atributos.

Porém no Python também temos a opção em definir atributos de um objeto sem precisar declarar na própria classe, tornando a sua criação bem dinâmico, como podemos ver a seguir:

```
class Funcionario:
    pass

func1 = Funcionario()
func1.nome = 'Fernando'
func1.sobrenome = 'Spelling'
```

E isso funciona pois em Python os objetos são **DINAMICOS**!! Ou seja, podemos adicionar atributos em qualquer momento, pois esses objetos utilizam um dicionário interno **(__dict__)** para armazenar seus dados!

Porém mesmo funcionando dessa forma, não é recomendável a sua utilização, pois apresenta as seguintes desvantagens:

- O atributo só existe nesse objeto, não em todos.
- Pode causar erros difíceis de rastrear.
- Não é visível ao ler a classe, portanto difícil de manter.

Criando construtores

Chegou o momento de falarmos desse recurso importantíssimo, que são os **construtores**!! E como já devemos saber os **construtores** de uma classe são executados sempre ao instanciar um novo objeto!!

E no **python** os construtores é definido pela função **_init_()**, como podemos ver a seguir do nosso exemplo da classe de funcionário.

```python
class Funcionario:
    nome: str
    sobrenome: str

    def __init__(self, nome: str, sobrenome: str):
        self.nome = nome
        self.sobrenome = sobrenome

    def imprimir(self):
        print(f'{self.nome} {self.sobrenome}')
```

Um detalhe importante a destacar é que todos os métodos de uma classe em **python**, incluindo o método de construtor, o primeiro parâmetro que é obrigatório a declarar é o **self**. E ele basicamente define que queremos trabalhar diretamente com ele mesmo, ou seja, que queremos trabalhar com a **própria instância** daquele objeto.

Fazendo uma analogia com outras linguagens de programação, seria semelhante ao utilizar o **this**.

Agora vamos instanciar um objeto da classe **Funcionario**, passando os valores já no construtor, e em seguida executaremos o método **imprimir()**.

```python
func = Funcionario('Fernando', 'Spelling')
func.imprimir()
```

Vale destacar também que essa é a maneira mais comum em chamarmos um método de um objeto em **python**, porém temos uma outra maneira em realizar, que por curiosidade é maneira como o Python realiza por padrão por **“debaixo dos panos”!**

```python
Funcionario.imprimir(func)
```

Veja que por padrão o **python** define a execução de um método diretamente pelo **nome da classe**, passando a **instância** no objeto como o **primeiro parâmetro**!! Por isso que quando criamos métodos de uma classe, precisamos definir o **self** como primeiro parâmetro!!