✨ Funções ✨

> Parâmetros, Retorno, Parâmetro Padrão, Parâmetro Arbitrários e Tipos de Dados

Nesta aula, vamos aprender sobre funções em Python. Funções são blocos de código que realizam uma tarefa específica e podem ser reutilizados em diferentes partes do programa. Vamos explorar conceitos importantes como parâmetros, retorno de valores, parâmetros padrão, tipos de parâmetros, e como especificar o tipo de dados de um parâmetro.

💡 O que é uma Função? 💡

Uma função é um bloco de código que pode ser chamado várias vezes. Elas são usadas para organizar o código em partes menores e mais fáceis de entender. Em Python, você define uma função com a palavra-chave `def`.

```
def saudacao():
    print("Olá, mundo!")
```

O código acima define uma função chamada `saudacao`. Quando chamamos `saudacao()`, ela executa o código dentro dela, que neste caso imprime "Olá, mundo!" na tela.

⚡ Parâmetros de Funções ⚡

Parâmetros são informações que passamos para a função para que ela faça algo com esses dados. Eles são colocados entre parênteses após o nome da função.

```
def saudacao(nome):
    print(f"Olá, {nome}!")
```

A função agora recebe um parâmetro chamado `nome`. Quando chamamos `saudacao("Ana")`, a função exibe "Olá, Ana!". O valor de `nome` é substituído pelo valor que passamos ao chamar a função.

⚡ Retorno de Valores ⚡

Algumas funções realizam um cálculo ou tarefa e nos devolvem um valor. Para isso, usamos a palavra-chave `return`.

```
def soma(a, b):
    return a + b
```

A função `soma` recebe dois parâmetros (`a` e `b`) e retorna o resultado da soma deles. Quando chamamos `soma(3, 4)`, o valor retornado será 7.

```
resultado = soma(3, 4)
print(resultado)  # Exibe 7
```

⚡ Parâmetros Padrão ⚡

Você pode definir um valor padrão para um parâmetro. Isso significa que, se o valor não for passado quando a função for chamada, o parâmetro usará o valor padrão.

```
def saudacao(nome="amigo"):
    print(f"Olá, {nome}!")
```

Se chamarmos `saudacao()` sem passar um nome, a função exibirá "Olá, amigo!". Se passarmos um nome, como `saudacao("Maria")`, a função usará o valor fornecido.

⚡ Tipos de Dados de Parâmetros ⚡

Em Python, podemos especificar os tipos de dados esperados para os parâmetros. Isso ajuda a tornar o código mais claro e legível, além de ajudar o programador a evitar erros. No entanto, o Python não exige que os tipos sejam seguidos estritamente — eles servem mais como uma dica.

```python
def somar(a: int, b: int) -> int:
    return a + b
```

O código acima define que a função `somar` espera dois parâmetros do tipo `int` (inteiro) e retorna um valor do tipo `int`. A anotação `-> int` indica o tipo de dado retornado pela função.

Observação: As anotações de tipo são opcionais, mas tornam o código mais legível e ajudam a evitar erros.

⚡ Parâmetros Arbitrários ⚡

Às vezes, você não sabe quantos argumentos serão passados para a função. Você pode usar o símbolo * para aceitar múltiplos argumentos.

```
def soma_todos(*numeros: int) -> int:
    return sum(numeros)
```

A função `soma_todos` pode receber qualquer número de parâmetros, e todos eles são somados usando a função `sum()`.

```
print(soma_todos(1, 2, 3, 4))  # Exibe 10
```

📋 Resumo da Aula 📋

- **`Função`**: Um bloco de código reutilizável que realiza uma tarefa.
- **`Parâmetro`**: Valores que passamos para a função para personalizar seu comportamento.
- **`Retorno`**: Usamos `return` para devolver um valor de uma função.
- **`Parâmetro`** Padrão: Um valor que a função usará se nenhum valor for passado.
- **`Tipos`** de Dados de Parâmetros: Indicam que tipos de valores a função espera.
- **`Parâmetros`** Arbitrários: Permite múltiplos argumentos usando `*`.