Funções Python

As **funções** são blocos de código reutilizáveis que realizam uma **tarefa específica**.

Elas permitem dividir um programa em **partes menores**, tornando o código mais **organizado e reutilizável**.

💡 **Benefícios das funções:**

- Evitam **repetição de código**
- Melhoram a **legibilidade**
- Facilitam a **manutenção e escalabilidade**

1️⃣ Como Criar uma Função?

✔️ Usando `def` para definir uma função

```
def saudacao():
    print("Olá, seja bem-vindo!")

saudacao()
# Saída: Olá, seja bem-vindo!
```

✔️ Funções com parâmetros

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

saudacao("Alice")
# Saída: Olá, Alice!
```

✔️ Função com retorno (`return`)

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

resultado = soma(5, 3)
print(resultado)
# Saída: 8
```

2️⃣ Parâmetros e Argumentos

✅ Parâmetros com valor padrão

```
def mensagem(texto="Olá!"):
    print(texto)

mensagem()
# Saída: Olá!
mensagem("Bem-vindo ao Python!")
# Saída: Bem-vindo ao Python!
```

✅ Passando múltiplos argumentos (`*args`)

```
python
CopyEdit
def soma_total(*numeros):
    return sum(numeros)

print(soma_total(1, 2, 3, 4, 5))
# Saída: 15
```

✅ Argumentos nomeados (`**kwargs`)

```
def dados_pessoais(**info):
    for chave, valor in info.items():
        print(f"{chave}: {valor}")

dados_pessoais(nome="Alice", idade=25, cidade="São Paulo")
# Saída:
```

```
# nome: Alice
# idade: 25
# cidade: São Paulo
```

3️⃣ Escopo de Variáveis

As variáveis podem ter **escopo local ou global**.

✅ **Variável local (só existe dentro da função)**

```
def exemplo():
    x = 10  # Variável local
    print(x)

exemplo()
# Saída: 10
print(x)  # Erro! x não existe fora da função
```

✅ **Variável global**

```
x = 5  # Variável global

def modificar():
    global x  # Modifica a variável global
    x = 10

modificar()
print(x)
# Saída: 10
```

4️⃣ Funções Anônimas (`lambda`)

As **funções lambda** são funções curtas e sem nome, úteis para operações simples.

```
quadrado = lambda x: x ** 2
print(quadrado(4))
```

```
# Saída: 16
```

Outra aplicação comum:

```
soma = lambda a, b: a + b
print(soma(2, 3))
# Saída: 5
```

5️⃣ Funções Dentro de Funções (Funções Aninhadas)

```
def externa():
    print("Função externa")

    def interna():
        print("Função interna")

    interna()

externa()
# Saída:
# Função externa
# Função interna
```

6️⃣ Funções Recursivas

Funções **recursivas** são aquelas que chamam a si mesmas para resolver problemas menores.

Exemplo: **Fatorial de um número**

```
def fatorial(n):
    if n == 1:
        return 1
    return n * fatorial(n - 1)
```

```
print(fatorial(5))
# Saída: 120
```

📌 Resumo Rápido

- 🔷 `def` define uma função
- 🔷 Funções podem receber **parâmetros e argumentos**
- 🔷 `return` retorna valores
- 🔷 `lambda` cria funções anônimas
- 🔷 Funções podem ser **aninhadas ou recursivas**