# Sintaxe Python

Funções

# Introdução

- As funções são blocos de código nomeados que realizam uma tarefa específica.
- Elas são fundamentais para organizar e reutilizar código de forma eficiente.

# Definição de Funções

- Uma função em Python é um bloco de código que realiza uma tarefa específica quando chamado.
- Elas ajudam a modularizar o código, tornando-o mais legível e fácil de manter.

## Definição de Funções

- As funções também promovem a reutilização de código, pois podem ser chamadas várias vezes de diferentes partes do programa.

# Estrutura de uma Função

- Em Python, as funções são definidas usando a palavra-chave **def**, seguida pelo **nome da função** e **parênteses** que podem conter **parâmetros**.
- O corpo da função é indentado e contém o código a ser executado quando a função é chamada.

```python
def funcao(): #Definição do nome da função
  #conteúdo da função
funcao() #chamada da função

#-----------EXEMPLO PRÁTICO--------------

def soma(numero_1, numero_2): #Definição do nome da função
  return numero_1 + numero_2
soma(15,14) #Resultado 29
```

# Parâmetros de Função

- Os parâmetros permitem que as funções aceitem dados de entrada.
  - Exemplos:
    - `def soma(num_1,num2):`
    - `def horaAtual():`
    - `def nomeCompleto(nome, sobrenomes):`

# Parâmetros de Função

- Existem três tipos principais de parâmetros em Python:
  - **Posicionais**: são passados na ordem em que são definidos.
  - **Nomeados**: são especificados explicitamente pelo nome.
  - **Padrão**: têm valores predefinidos caso não sejam fornecidos pelo usuário.

# Parâmetros Posicionais

- Os parâmetros posicionais são os mais comuns em funções Python.
- Eles são passados na ordem em que são definidos na assinatura da função.
- O número de argumentos passados deve corresponder ao número de parâmetros na definição da função.

# Parâmetros Posicionais

```
def saudacao(nome, mensagem):
    return f"{mensagem}, {nome}!"

print(saudacao("Ana", "Olá"))  # Saída: Olá, Ana!
```

# Parâmetros Nomeados

- Os parâmetros nomeados são especificados pelo nome do parâmetro, permitindo que os argumentos sejam passados em qualquer ordem.
- Isso oferece mais clareza e flexibilidade ao chamar a função.

# Parâmetros Nomeados

-

```
def saudacao(nome, mensagem):
    return f"{mensagem}, {nome}!"

print(saudacao(mensagem="Olá", nome="João"))  # Saída: Olá, João!
```

# Parâmetros Padrão

- Os parâmetros padrão têm um valor predefinido, que é usado quando nenhum valor é especificado durante a chamada da função.
- Eles permitem que a função seja chamada com menos argumentos do que o número total de parâmetros definidos.

# Parâmetros Padrão

```python
def saudacao(nome, mensagem="Olá"):
    return f"{mensagem}, {nome}!"

print(saudacao("Maria"))  # Saída: Olá, Maria!
```

# Retorno de Função

- O retorno de uma função é feito usando a palavra-chave **`return`**.
- Ele permite que a função envie um resultado de volta para o local onde foi chamada.
- As funções podem retornar qualquer tipo de dado, incluindo números, strings, listas e até outras funções.

# Escopo de Variáveis

- O escopo de uma variável determina onde ela pode ser acessada dentro do código.
- As variáveis definidas dentro de uma função têm escopo local e só podem ser acessadas dentro dessa função.
- Variáveis definidas fora de todas as funções têm escopo global e podem ser acessadas de qualquer lugar do código.

# Escopo de Variáveis

```
def minha_funcao():
    x = 10  # Variável local
    print("Variável local x:", x)

minha_funcao()
# Saída: Variável local x: 10

# Tentar acessar x fora da função resultará em um erro
# print(x)  # Isso causaria um NameError
```

# Escopo de Variáveis

```
y = 20  # Variável global

def minha_funcao():
    print("Variável global y:", y)

minha_funcao()
# Saída: Variável global y: 20
```

# Escopo de Variáveis

```
z = 30  # Variável global

def minha_funcao():
    z = 40  # Variável local com o mesmo nome
    print("Variável local z:", z)

minha_funcao()
# Saída: Variável local z: 40

print("Variável global z:", z)
# Saída: Variável global z: 30
```