FUNÇÕES EM PYTHON: CONCEITOS, EXEMPLOS E IMPORTAÇÃO DE CÓDIGOS EXTERNOS

OBJETIVO DA APRESENTAÇÃO

- Entender o que são funções em Python
- Aprender a criar e usar funções simples
- Ver exemplos práticos de funções
- Conhecer formas de chamar scripts externos: `runpy`, `exec`, `subprocess`
- Discutir problema de acentuação e como resolver
- Saber quando usar cada forma de execução externa

O QUE É UMA FUNÇÃO?

- Bloco de código que executa uma tarefa específica
- Pode receber dados de entrada (parâmetros)
- Pode retornar um resultado
- Ajuda a organizar, reaproveitar e facilitar manutenção do código

EXEMPLO BÁSICO DE FUNÇÃO (MÉDIA DE NOTAS)

```
def calcula_media(n1, n2, n3, n4):
    return (n1 + n2 + n3 + n4) / 4


notas = [7, 8, 9, 6]
media = calcula_media(*notas)
print(f'Média: {media:.2f}')
```

Explicação:

- **Criamos uma função que calcula a média das 4 notas**
- **Passamos as notas e imprimimos o resultado formatado**

FUNÇÕES PARA MAIOR E MENOR NOTA

```
def maior_nota(notas):
    return max(notas)

def menor_nota(notas):
    return min(notas)
```

```
notas = [7, 8, 9, 6]
print(f'Maior nota: {maior_nota(notas)}')
print(f'Menor nota: {menor_nota(notas)}')
```

EXERCÍCIOS EM FUNÇÕES: MENU COM OPÇÕES

Exercício 5: Converter metros para centímetros

Exercício 6: Calcular área do círculo

Exercício 7: Calcular área do quadrado

Cada exercício em uma função separada

EXEMPLO FUNÇÃO EXERCÍCIO 5 (METROS PARA CENTÍMETROS)

```
def metros_para_centimetros():


    metros = float(input("Digite metros: "))
    print(f"{metros} metros equivalem a {metros * 100:.2f} cm")
```

EXECUTANDO EXERCÍCIOS EM UM MENU (FUNÇÕES)

```
def menu():
    while True:
        print("1 - Ex 5\n2 - Ex 6\n3 - Ex 7\n0 - Sair")
        op = input("Escolha: ")
        if op == '1': metros_para_centimetros()
        elif op == '2': area_circulo()
        elif op == '3': area_quadrado()
        elif op == '0': break
        else: print("Opção inválida")
```

CHAMAR SCRIPTS EXTERNOS: FORMAS COMUNS

```
runpy.run_path("arquivo.py")
exec(open("arquivo.py").read())
subprocess.run(["python", "arquivo.py"])
```

RUNPY.RUN_PATH()

- ✓ Executa script como se fosse um módulo
- ✓ Isola o contexto do script
- ✓ Pode apresentar problemas de codificação e saída no terminal
- ✓ Não roda novo processo, execut

```
python

import runpy
runpy.run_path("6.py")
```

EXEC()

- ✓ Executa código Python diretamente de uma string ou arquivo
- ✓ Código roda dentro do processo atual, sem isolamento
- ✓ Risco de sobrepor variáveis locais/globais
- ✓ Mais suscetível a erros e problemas de manutenção

```
python

exec(open("6.py").read())
```

SUBPROCESS.RUN()

- ✓ Roda o script como um processo separado
- ✓ Entrada e saída funcionam normalmente no terminal
- ✓ Mantém ambiente do terminal (como codificação UTF-8)
- ✓ Recomendado para executar scripts externos que usam input/output

```python
import subprocess
subprocess.run(["python", "6.py"])
```

POR QUE USAR SUBPROCESS EM VEZ DE RUNPY OU EXEC?

- ✓ Garante que a acentuação funcione corretamente no terminal
- ✓ Evita conflito entre namespaces dos scripts
- ✓ Facilita o controle do processo e saída padrão
- ✓ Maior compatibilidade entre sistemas operacionais

PROBLEMAS COMUNS COM ACENTUAÇÃO

- Codificação errada do arquivo fonte
- Terminal não configurado para UTF-8 (Windows)
- Falta de linha `# -*- coding: utf-8 -*-` no topo do script
- Solução: salvar arquivo em UTF-8 + `chcp 65001` no terminal Windows + usar `subprocess`

EXEMPLO PRÁTICO DE ACENTUAÇÃO

python

```python
# -*- coding: utf-8 -*-
raio = float(input('Informe o raio do círculo: '))
area = 3.14 * (raio ** 2)
print(f'Área do círculo com raio {raio}: {area:.2f}')
```

DICAS

- ✓ Use funções para organizar código
- ✓ Separe exercícios em arquivos/funções distintas
- ✓ Prefira `subprocess` para chamar scripts externos
- ✓ Sempre garanta codificação UTF-8 nos arquivos e terminal
- ✓ Teste scripts individualmente antes de integrar no menu