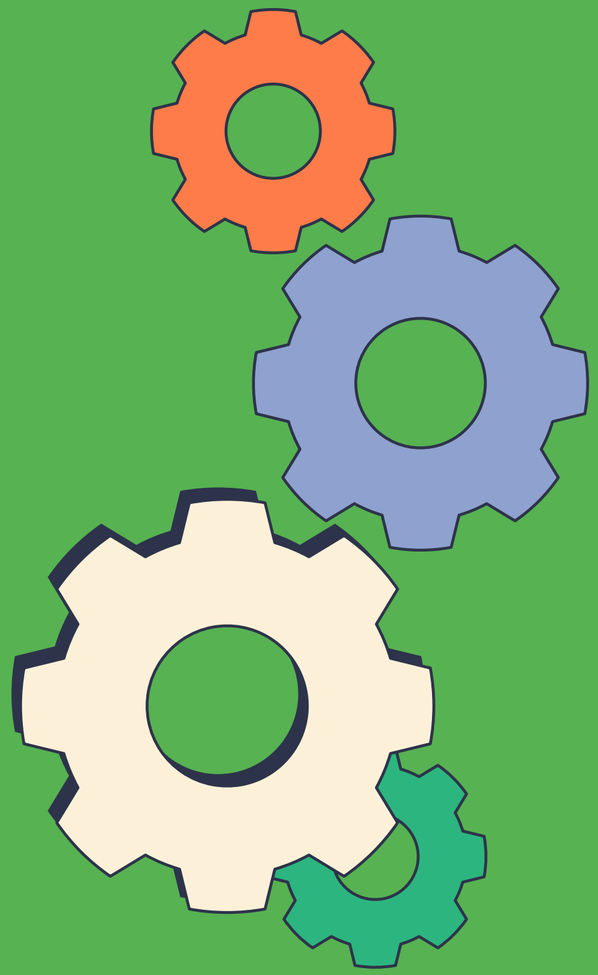

Aula 5: Funções

FUNDAMENTOS DE PROGRAMAÇÃO

CodeRoots

O que são funções

Conforme os programas crescem, o código pode ficar repetitivo e difícil de entender.

As funções ajudam a dividir o programa em partes menores e reutilizáveis.

- Uma função é um bloco de código que executa uma tarefa específica.
- Elas permitem reutilizar código, organizar melhor o programa e facilitar a manutenção.
- Você pode chamar uma função várias vezes, sempre que precisar daquela ação.

Em resumo:

Funções servem para dividir o problema em partes menores, deixando o programa mais limpo, fácil de entender e reaproveitável.

Exemplo do dia a dia:

“Ligar o liquidificador” é como chamar uma função: ele executa uma tarefa e para quando termina.

Estrutura básica de uma função

Sintaxe (Como deve ser escrito):

```
def nome_da_funcao():
    # bloco de código
```

Exemplo:

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

Chamando a função:

```
saudacao()
```

>

- A palavra def indica que estamos definindo uma função.
- O nome da função vem logo depois.
- Os parênteses () indicam a chamada da função.

Funções com parâmetros

- Parâmetros são valores que a função recebe para usar dentro dela.
- Permitem personalizar o que a função faz.

Exemplo:

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

Chamando:

```
saudacao("Matheus")
saudacao("Miura")
```

>

- O valor "Matheus" é enviado para o parâmetro nome.
- Dentro da função, o nome é usado no print().

Funções que retornam valores

- Algumas funções não apenas executam uma ação, mas também retornam um resultado.
- Isso é feito com a palavra return.

Exemplo:

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

Chamando:

```
total = soma(3, 5)
print("A soma é:", total)
```

>

- A função faz o cálculo e envia o resultado de volta com return.
- Esse valor pode ser guardado em uma variável.

Por que usar funções

- Organização: o código fica dividido em partes pequenas.
- Reutilização: você escreve uma vez e usa várias.
- Leitura fácil: cada função tem um nome que descreve o que ela faz.
- Facilidade na manutenção: se precisar mudar algo, altera só dentro da função.

Exemplo:

Um programa de cadastro pode ter funções como `cadastrar_usuario()`, `mostrar_menu()`, `salvar_dados()`, etc.

Parâmetros e retornos juntos

Exemplo completo:

```python
def media(n1, n2):
    m = (n1 + n2) / 2
    return m

nota_final = media(8, 7)
print("Média final:", nota_final)
```

- A função recebe dois números, calcula a média e retorna o resultado.
- A variável nota_final guarda o valor retornado.

Boas práticas ao criar funções

- Use nomes claros e descritivos.
- Cada função deve fazer apenas uma tarefa.
- Sempre teste a função isoladamente antes de integrar ao programa.
- Use comentários (#) para explicar o que ela faz.

Exemplo de bom nome:

```
def calcular_desconto(preco, percentual):
```

Muito melhor do que `def func1(p, x):`

Encerramento do módulo!

Você concluiu o módulo de **Fundamentos da Programação!**

Nesta jornada, você aprendeu:

1. **O que é programação**
2. **Variáveis e tipos de dados**
3. **Estruturas condicionais**
4. **Laços de repetição**
5. **Funções**

A partir daqui, você já entende a base de toda lógica de programação , pronto para explorar estruturas de dados, algoritmos e até linguagens mais avançadas.

Parabéns!

Obrigado por participar do curso de Fundamentos da Programação!

CodeRoots