Funções

Lógica de Programação

Introdução às Funções

Funções são blocos reutilizáveis de código projetados para executar uma tarefa específica.

Neste material, abordaremos:

1. Definição e benefícios das funções.
2. Declaração e chamada de funções.
3. Funções com parâmetros e retorno.
4. Escopo de variáveis.
5. Exemplos práticos e exercícios.

O que são Funções?

Funções são blocos de código nomeados que podem ser executados sob demanda.

Benefícios:

- Reutilização de código.
- Modularidade.
- Melhor organização e legibilidade.
- Redução de erros e facilidade de manutenção.

Estrutura de uma Função

Uma função típica consiste em:

- **Nome:** Identifica a função.
- **Parâmetros (opcional):** Dados de entrada.
- **Bloco de código:** Executa a tarefa.
- **Retorno (opcional):** Devolve um resultado.

Sintaxe (Python):

```
def nome_funcao(parâmetros):
    # bloco de código
    return resultado
```

Declaração de Funções

Declaração cria uma função, mas não a executa.

Exemplo em Python:

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

Exemplo em Java:

```
void saudacao() {
  System.out.println('Olá, mundo!');
}
```

Chamada de Funções

Executa o bloco de código definido na função.

Exemplo em Python:

- saudacao()

Exemplo em Java:

- saudacao();

Funções com Parâmetros

- Permitem enviar dados para a função.

- Exemplo em Python:
- def saudacao(nome):
- print(f'Olá, {nome}!')

- saudacao('João') # Saída: Olá, João!

Funções com Retorno

Permitem devolver um resultado para quem chamou a função.

Exemplo em Python:

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

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

Escopo de Variáveis

Define onde uma variável é acessível no programa.

- Escopo local: Dentro de uma função.
- Escopo global: Fora de qualquer função.

Exemplo:

```
def exemplo():
    local = 'dentro'
global_var = 'fora'
```

Funções Recursivas

Chamam a si mesmas para resolver problemas menores do mesmo tipo.

Exemplo: Fatorial em Python

```
def fatorial(n):
  if n == 0:
    return 1
  return n * fatorial(n - 1)
fatorial(5)  # Resultado: 120
```

Função vs. Procedimento

- **Função:** Retorna um valor.
- **Procedimento:** Executa uma tarefa sem retornar valor.

Exemplo de Procedimento:

```
def imprimir_mensagem():
    print('Mensagem sem retorno')
```

Exemplo: Calculadora Simples

Função que soma dois números:

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

resultado = soma(10, 5)
print(f'Resultado: {resultado}')
```

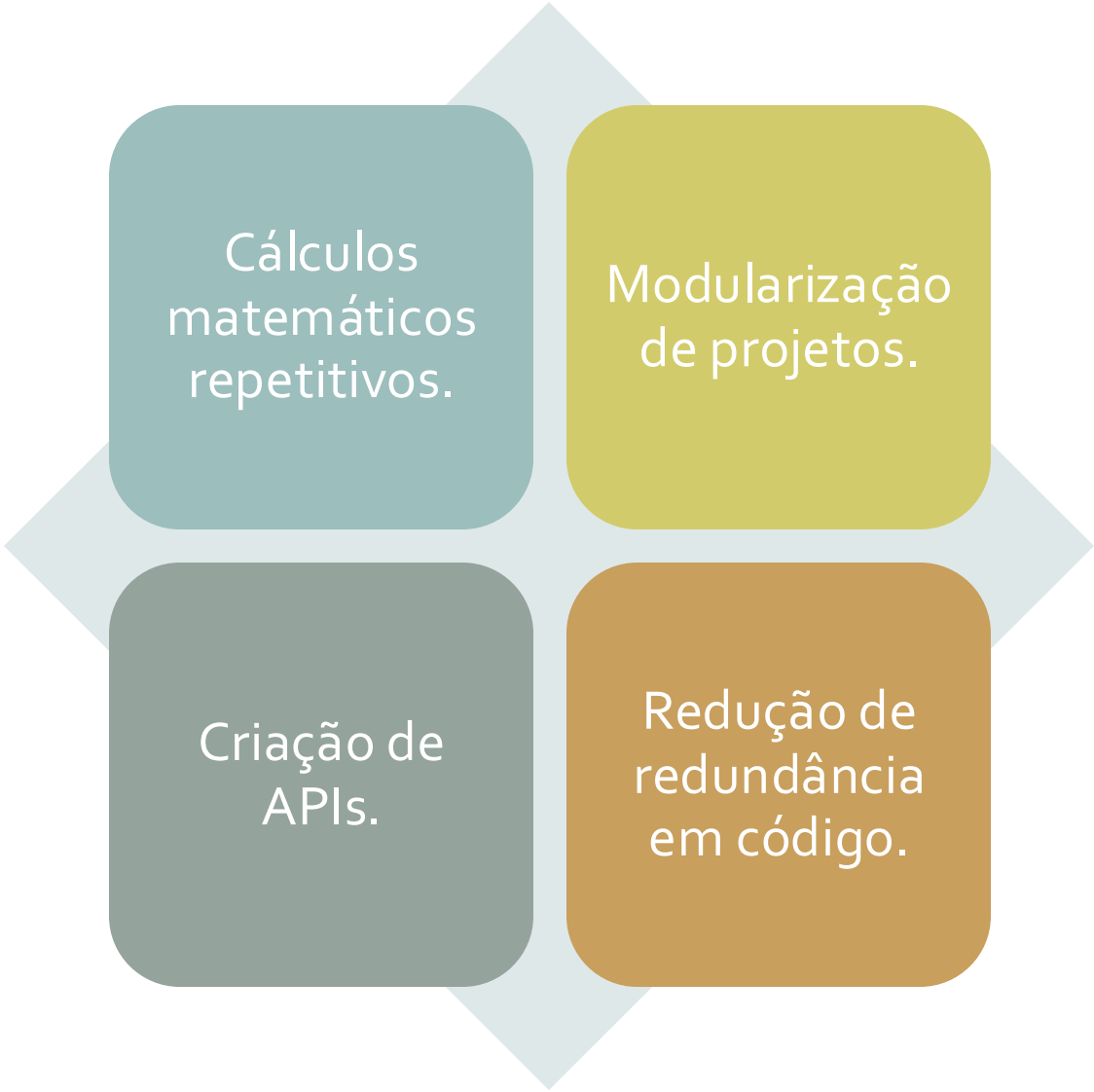
Exemplo: Verificação de Paridade

Função para verificar se um número é par:

```
def is_par(num):
    return num % 2 == 0

print(is_par(4))  # Saída: True
print(is_par(5))  # Saída: False
```

Aplicações Práticas das Funções



Exercícios de Fixação (Parte 1)

1. Crie uma função que calcule o dobro de um número.
2. Implemente uma função que retorne a média de três números.

Exercícios de Fixação (Parte 2)

3. Desenvolva uma função que determine se um número é primo.
4. Implemente uma função recursiva para calcular o n-ésimo número de Fibonacci.

Benefícios do Uso de Funções

- Melhoria da legibilidade do código.
- Reutilização em diferentes partes do programa.
- Redução de erros devido à modularidade.
- Simplificação da depuração e manutenção.

Revisão do Material

Neste material, aprendemos sobre:

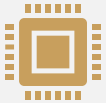
- Declaração e chamada de funções.
- Parâmetros e retorno.
- Escopo e recursão.
- Diferenças entre funções e procedimentos.

Aplicações no Mundo Real

- Construção de bibliotecas e APIs.

- Desenvolvimento de software modular.

- Implementação de algoritmos complexos.



- Redução de redundância em sistemas corporativos.

Referências

- 1. Apostila de Lógica de Programação - Maromo.
- 2. Exemplos práticos em Python e Java.
- 3. Documentação oficial das linguagens.