Aula 4: Funções e Modularização em Python

Objetivo da Aula:
Ao final desta aula, os alunos serão capazes de:

- recapitular os avisos padrões das atividades
-
- Entender a importância e a utilidade das funções em Python.
- Declarar e utilizar funções, compreendendo passagem de parâmetros e retorno.
- Aplicar modularização em projetos de Python, organizando o código de forma eficiente e escalável.

Definição:

Funções são blocos de código projetados para executar tarefas específicas. Elas podem ser reutilizadas em diferentes partes de um programa para aumentar a eficiência e reduzir a redundância.

Por que usar funções?

- Reutilização
- Organização de código aprimorada
- Depuração e teste mais fáceis

Conteúdo Programático:

1. **Introdução às Funções:**
    - Definição de função como um bloco de código que realiza uma tarefa específica.
    - Reuso de funções em diferentes partes de um programa.

Exemplo:

```
def calcular_area(largura, altura):
    area = largura * altura
    return area

# Chamando a função
print(calcular_area(5, 3))
```

2. **Declaração de Funções e Passagem de Parâmetros:**
    - Elementos principais de uma função: nome, parâmetros, corpo e retorno.

Conceitos-chave:

A declaração de função envolve dar um nome a um bloco de código.

A passagem de parâmetros permite que funções recebam dados e operem neles.

Exemplo didático:

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

cumprimentar("Ana")
```

-

Exemplo de função com retorno:

```
def eh_par(numero):
    return numero % 2 == 0

numero = 4
if eh_par(numero):
    print(f"{numero} é par")
else:
    print(f"{numero} é ímpar")
```

3. **Modularização de Código:**
    - Explicação sobre a importância da modularização para a organização e manutenção do código.
    - Criação de módulos em Python e como importá-los em outros arquivos.
    - **Exemplo prático de estrutura de módulos:**

Arquivo `modulo_matematica.py`:

```
def somar(a, b):
    return a + b
def subtrair(a, b):
    return a - b
```

Arquivo `principal.py`:

```
from modulo_matematica import somar, subtrair from modulo_listas
import encontrar_maximo print(somar(10, 5)) lista = [3, 1, 4, 1, 5]
print(f"Máximo: {encontrar_maximo(lista)}")
```

modulo_listas.py:

```
def encontrar_maximo(lista):

    return max(lista)
```

4. **Algoritmos com Funções:**
    - Prática de funções que implementam algoritmos básicos, como encontrar o valor máximo em uma lista, calcular a média e ordenar uma lista.

Exemplo 1: Encontrar o máximo de uma lista

```
def encontrar_maximo(lista):
    maximo = lista[0]
    for numero in lista:
        if numero > maximo:
            maximo = numero
    return maximo

numeros = [3, 5, 7, 2, 8]
print(f"O maior número é: {encontrar_maximo(numeros)}")
```

Exemplo 2: Calcular a média de uma lista

```
def calcular_media(lista):
    return sum(lista) / len(lista)

numeros = [3, 5, 7, 2, 8]
print(f"A média dos números é: {calcular_media(numeros)}")
```

Atividades Práticas:

- **Exercício 1:** Implemente uma função que verifique se um número é primo.
- **Exercício 2:** Crie um módulo que contenha funções matemáticas (somar, subtrair, multiplicar, dividir) e outro módulo que trabalhe com listas (encontrar máximo, mínimo, média).
- **Exercício 3:** Organize um pequeno projeto modularizado que calcule a área de diferentes figuras geométricas.

Recursos Utilizados:

- Python (ambiente de programação).

- Slides da aula 4 (disponibilizados em formato PDF para consulta).
- Editor de código para prática dos exemplos.

Conclusão:

- Reforçar a importância de funções e modularização para criar códigos organizados, eficientes e reutilizáveis.
- Prática intensiva recomendada para fixar os conceitos abordados.