Differentiated Problem Solving

Aula 03 - FIAP

Prof. Jones Egydio

Funções essenciais

Setup Python

```
import matplotlib.pyplot as plt  # Usado para criar gráficos e visualizações
import numpy as np  # Numpy é uma biblioteca fundamental para operações matemáticas e manipulação de arrays
```

Função para visualização gráfica

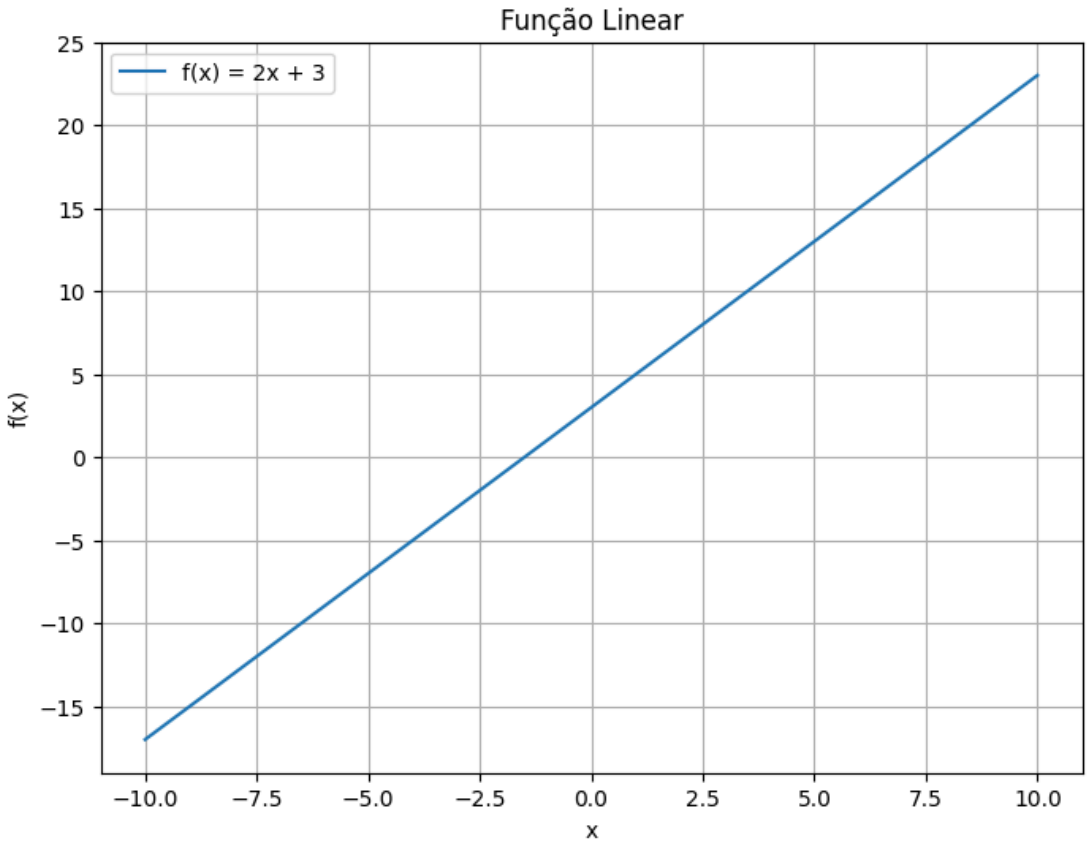
```
def plotar_funcao(func, x_range, titulo='', xlabel='x', ylabel='f(x)', legenda=''):
    x = np.linspace(x_range[0], x_range[1], 400)
    y = func(x)

    plt.figure(figsize=(8, 6))
    plt.plot(x, y, label=legenda or titulo)
    plt.title(titulo)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.grid(True)
    if legenda:
        plt.legend()
    plt.show()
```

Funções essenciais

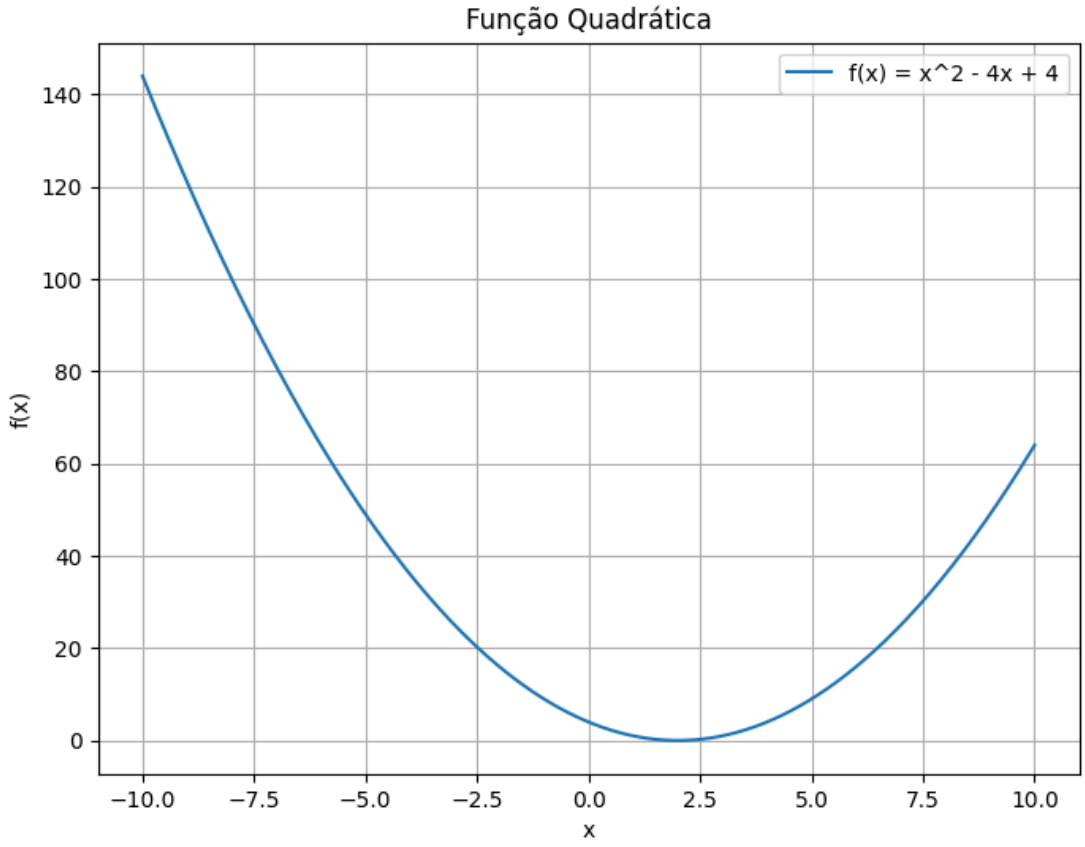
```
def funcao_linear(x):
    return 2*x + 3

def funcao_quadratica(x):
    return x**2 - 4*x + 4

def funcao_polinomial(x):
    return x**3 - 6*x**2 + 11*x - 6

def funcao_potencia(x):
    return x**0.5

def funcao_racional(x):
    # Definição para evitar divisão por zero
    return np.where(x == 3, np.nan, (x**2 - 9) / (x - 3))

def funcao_trigonometrica(x):
    return np.sin(x)

def funcao_exponencial(x):
    return np.exp(x)

def funcao_logaritmica(x):
    # Definição segura para evitar log de números não positivos
    return np.where(x > 0, np.log(x), np.nan)
```

Plotando os gráficos

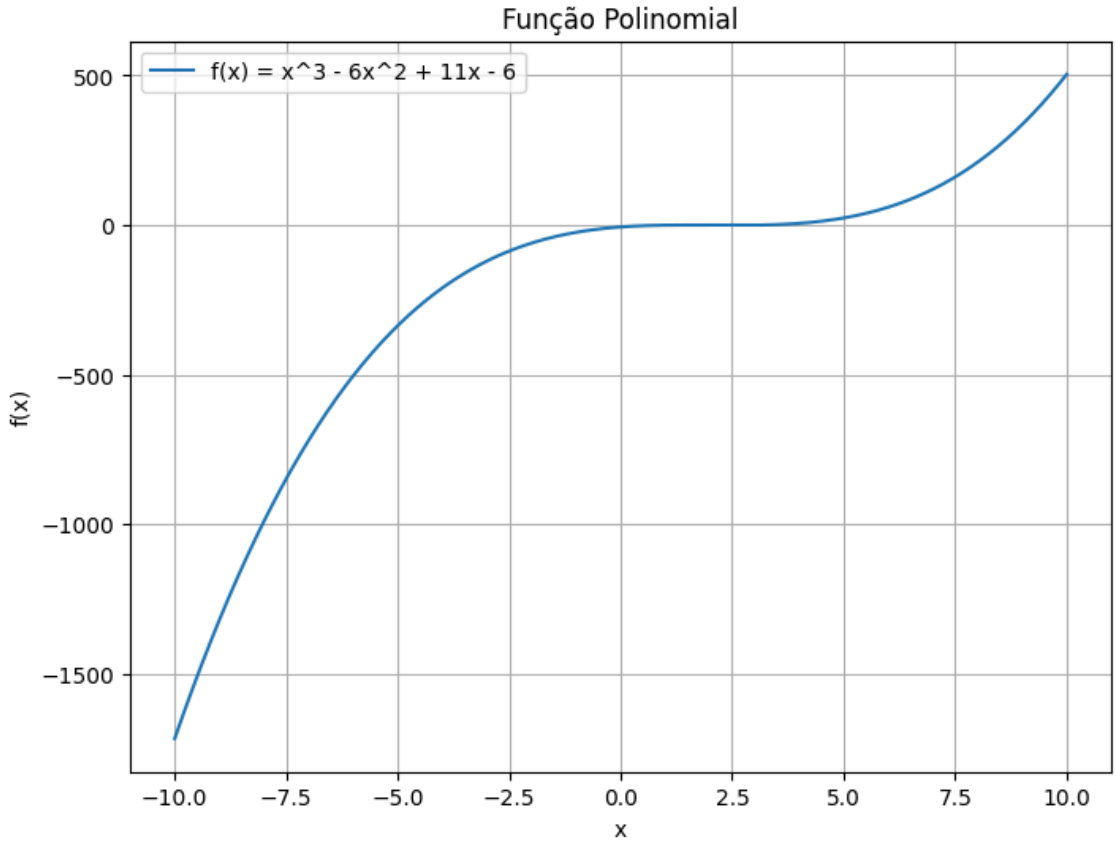
```
plotar_funcao(funcao_linear, [-10, 10], 'Função Linear', legenda='f(x) = 2x + 3')
```



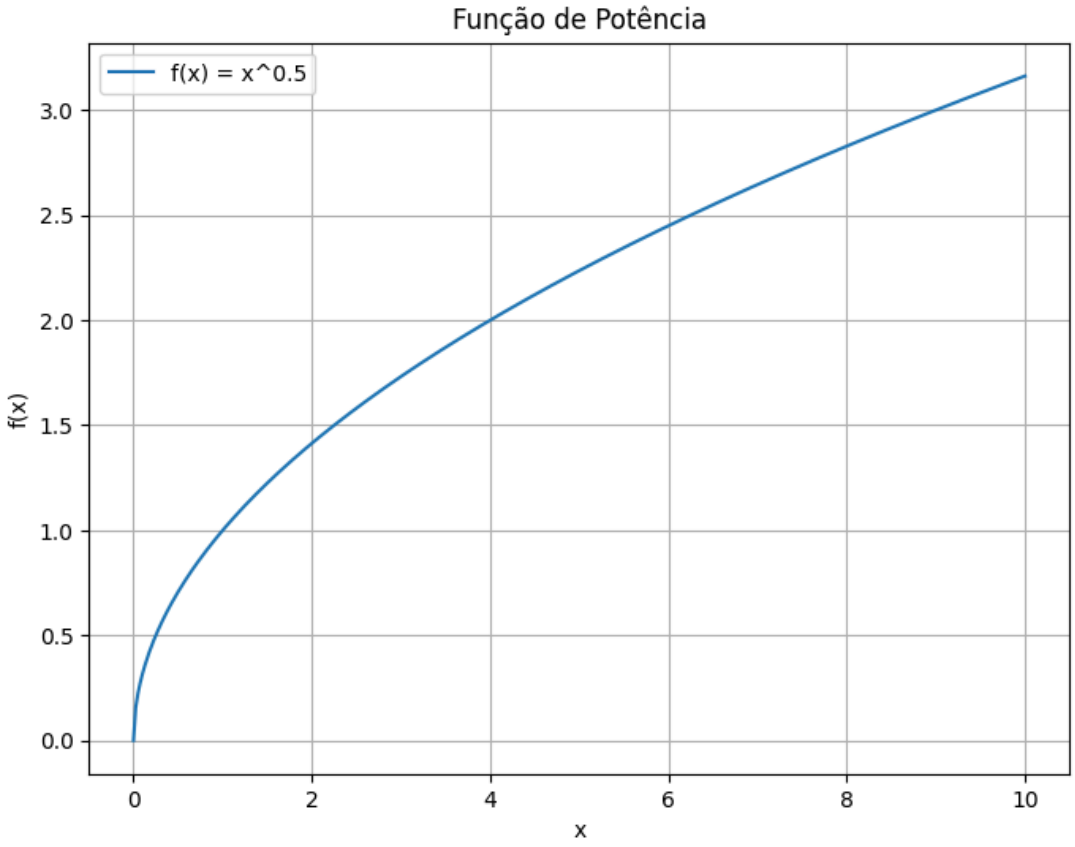

```
plotar_funcao(funcao_quadratica, [-10, 10], 'Função Quadrática', legenda='f(x) = x^2 - 4x + 4')
```



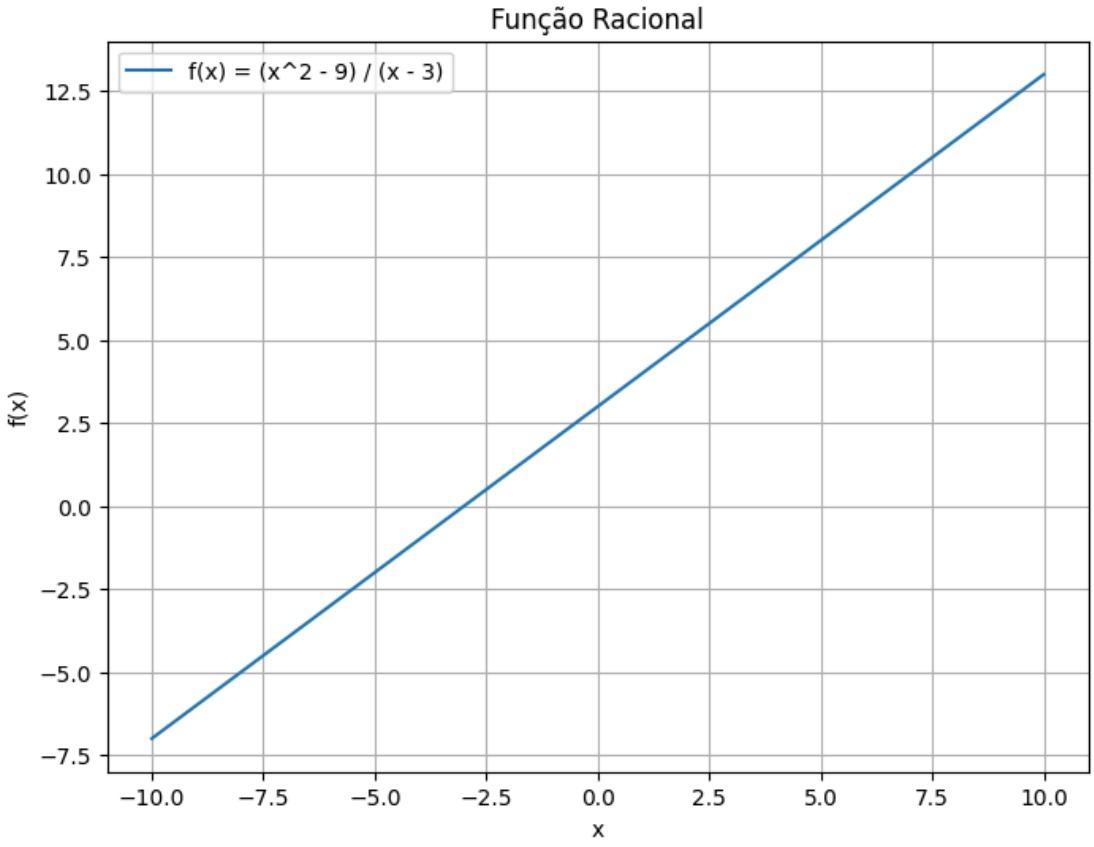

```
1 plotar_funcao(funcao_polinomial, [-10, 10], 'Função Polinomial', legenda='f(x) = x^3 - 6x^2 + 11x - 6')
```



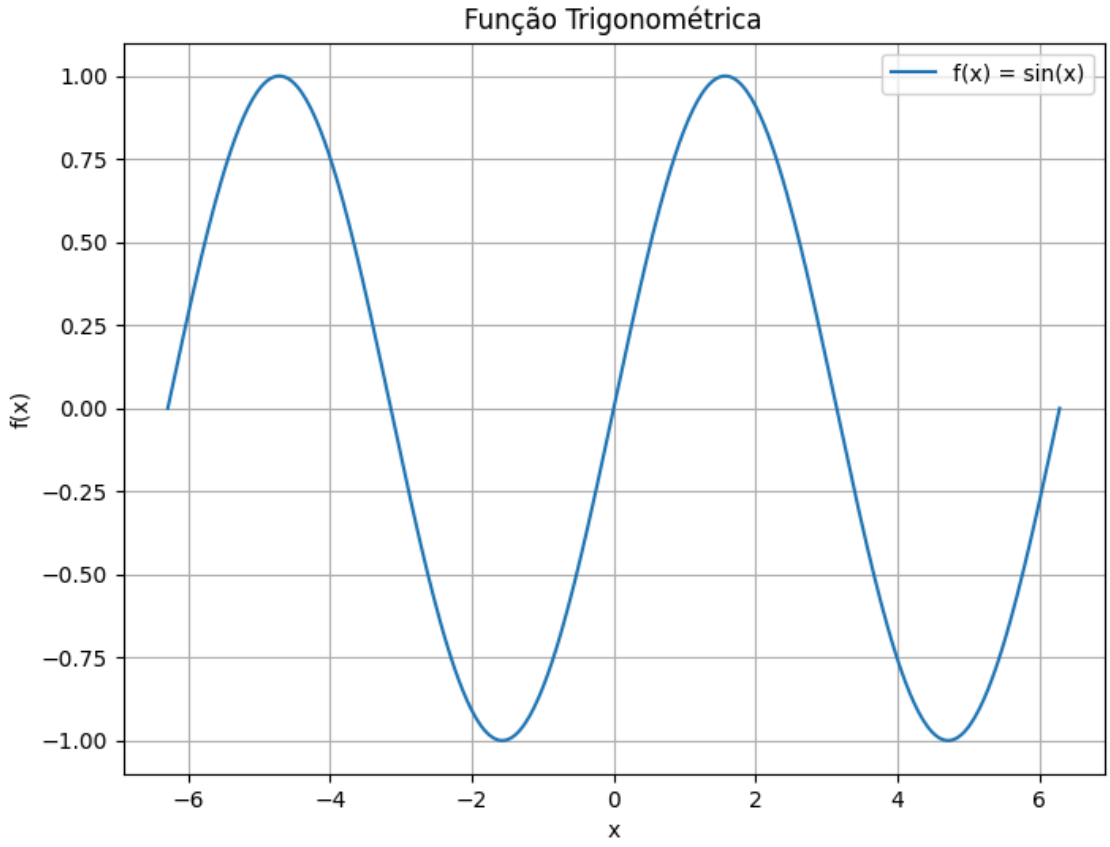

```
1 plotar_funcao(funcao_potencia, [0, 10], 'Função de Potência', legenda='f(x) = x^0.5')
```



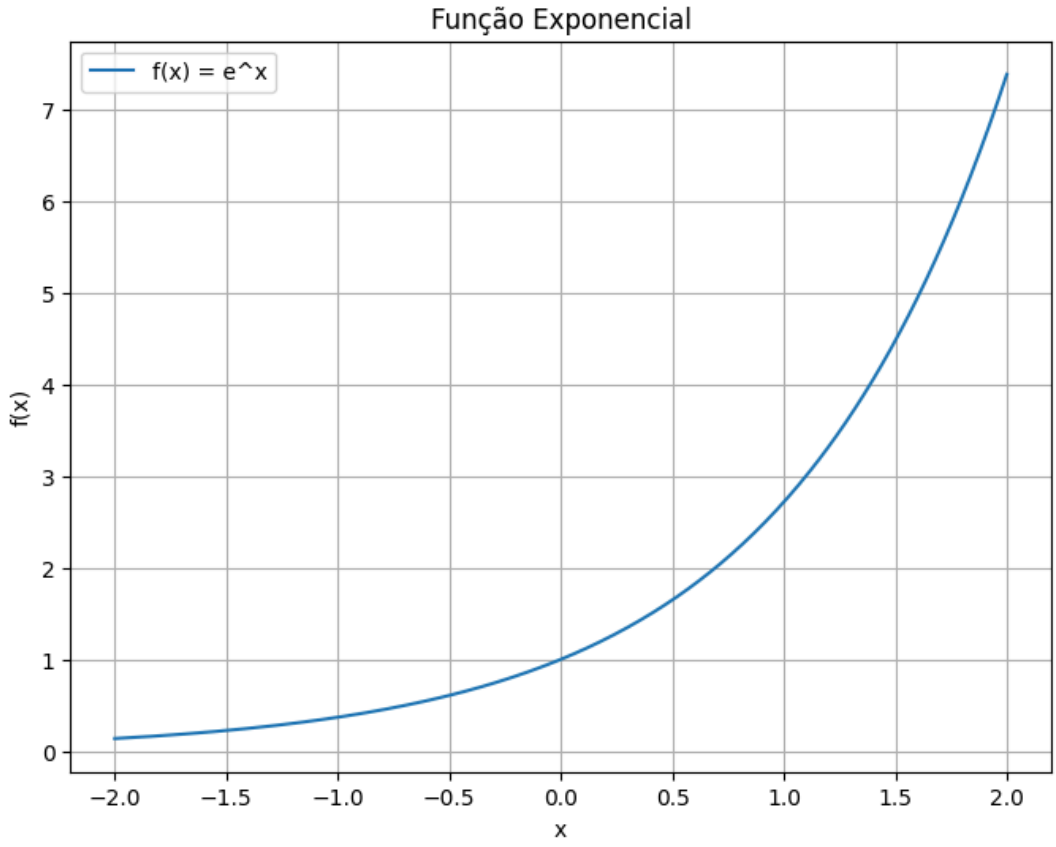

```
1 plotar_funcao(funcao_racional, [-10, 10], 'Função Racional', legenda='f(x) = (x^2 - 9) / (x - 3)')
```



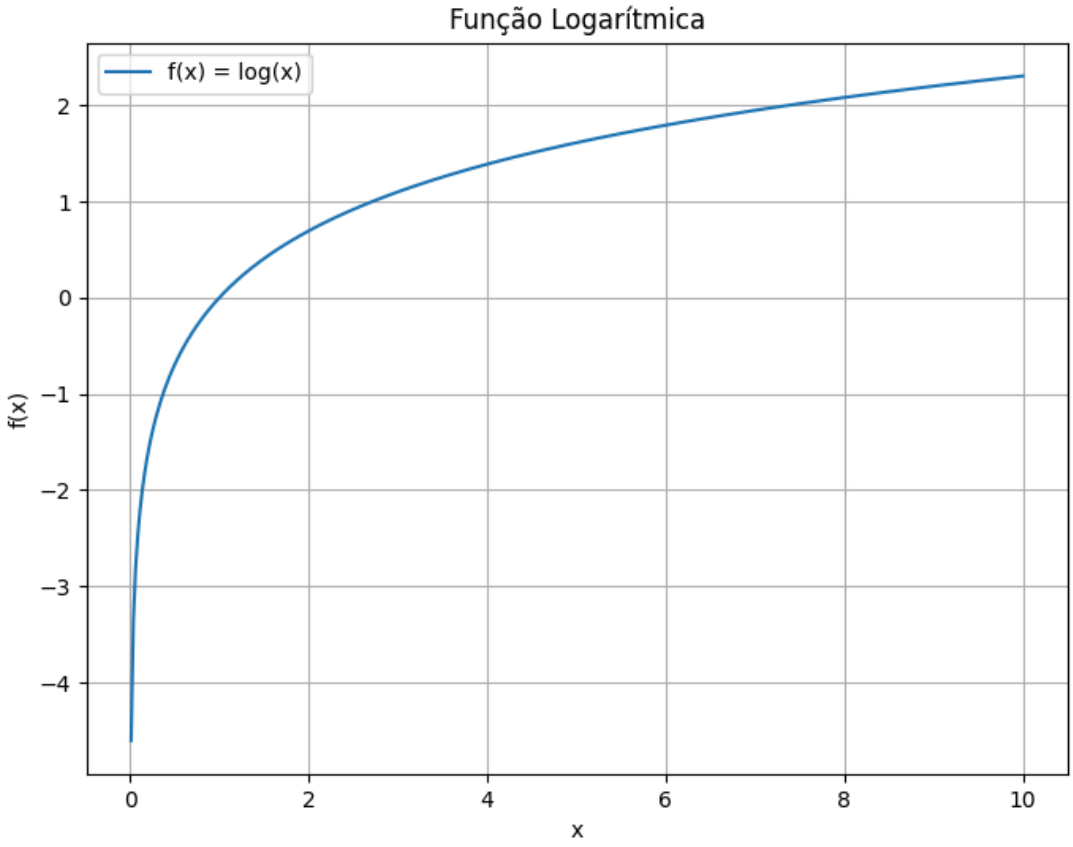

```
1 plotar_funcao(funcao_trigonometrica, [-2*np.pi, 2*np.pi], 'Função Trigonométrica', legenda='f(x) = sin(x)')
```




```
1 plotar_funcao(funcao_exponencial, [-2, 2], 'Função Exponencial', legenda='f(x) = e^x')
```




```
1 plotar_funcao(funcao_logaritmica, [0.01, 10], 'Função Logarítmica', legenda='f(x) = log(x)')
```

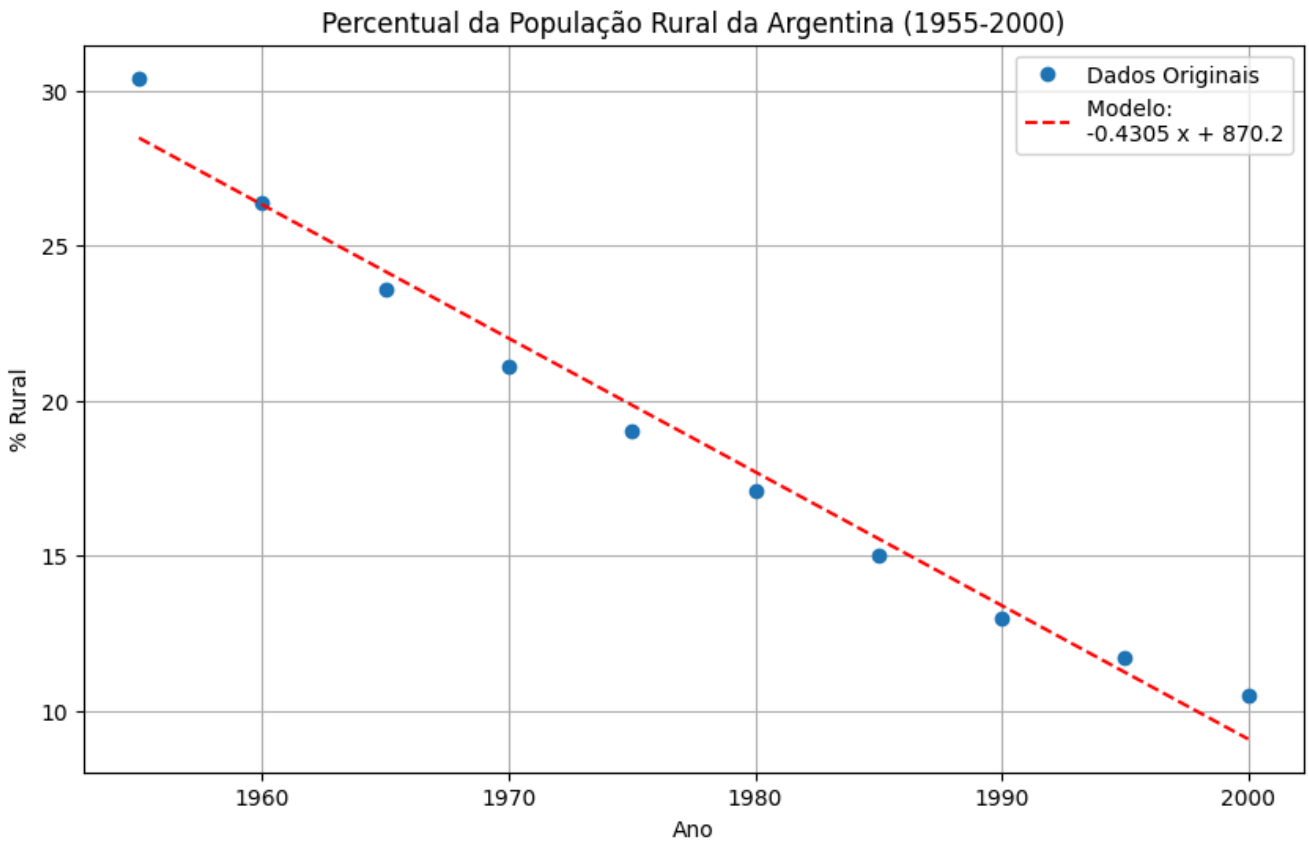


Exercícios

Ex01: A tabela mostra a porcentagem da população da Argentina que vivia em áreas rurais de 1955 a 2000. Encontre um modelo para os dados e utilize-o para estimar a porcentagem rural em 1988 e 2002.

Ano	% Rural	Ano	% Rural
1955	30,4	1980	17,1
1960	26,4	1985	15,0
1965	23,6	1990	13,0
1970	21,1	1995	11,7
1975	19,0	2000	10,5

```
# Dados fornecidos no problema
anos = np.array([1955, 1960, 1965, 1970, 1975, 1980, 1985, 1990, 1995, 2000])
percent_rural = np.array([30.4, 26.4, 23.6, 21.1, 19.0, 17.1, 15.0, 13.0, 11.7, 10.5])

# Ajustando um modelo linear
coeficientes = np.polyfit(anos, percent_rural, 1) # '1' significa uma linha reta
modelo = np.poly1d(coeficientes)

# Usando o modelo para estimar valores para 1988 e 2002
estimativa_1988 = modelo(1988)
estimativa_2002 = modelo(2002)

# Plotando os dados e a linha de melhor ajuste
plt.figure(figsize=(10, 6))
plt.plot(anos, percent_rural, 'o', label='Dados Originais')
plt.plot(anos, modelo(anos), 'r--', label=f'Modelo: {modelo}')
plt.title('Percentual da População Rural da Argentina (1955-2000)')
plt.xlabel('Ano')
plt.ylabel('% Rural')
plt.legend()
plt.grid(True)
plt.show()

(estimativa_1988, estimativa_2002)
```




```
(14.259272727272332, 8.231636363635971)
```

Explicando:

1. `coeficientes = np.polyfit(anos, percent_rural, 1)` : Esta linha está usando a função `polyfit` do Numpy para ajustar um modelo polinomial de grau 1 (uma linha reta) aos dados fornecidos. Os argumentos da função são:
   - `anos` : o array do Numpy contendo os anos dos dados.
   - `percent_rural` : o array do Numpy contendo as porcentagens correspondentes da população rural.
   - `1` : o grau do polinômio a ser ajustado. Neste caso, um polinômio de grau 1 corresponde a uma função linear, que tem a forma `y = mx + b`, onde `m` é o coeficiente angular (inclinação) e `b` é o intercepto y (o valor de `y` quando `x` é 0).

A função `polyfit` retorna os coeficientes `[m, b]` do polinômio que melhor se ajusta aos dados no sentido dos mínimos quadrados. Isso significa que a função tenta minimizar a soma dos quadrados das diferenças entre os valores observados e os valores ajustados pelo modelo.

2. `modelo = np.poly1d(coeficientes)` : Esta linha está criando um objeto polinomial a partir dos coeficientes obtidos pelo `polyfit` . A função `poly1d` é uma conveniência do Numpy que cria uma função polinomial que pode ser usada para calcular os valores de `y` para qualquer valor de `x` . Essencialmente, transforma os coeficientes numéricos em uma função representativa do modelo polinomial que pode ser chamada posteriormente, como `modelo(x)` .

   Com esse objeto `modelo` , você pode passar um valor de `x` (neste caso, um ano) e ele retornará o valor de `y` (neste caso, a porcentagem estimada da população rural) de acordo com o modelo linear ajustado. Isso é útil para fazer previsões com base no modelo, como você viu para estimar as porcentagens para os anos de 1988 e 2002.

Ex02: A chuva é essencial para crescimento das lavouras, no entanto, em demasiado, pode ocorrer o efeito reverso. Na Tabela 1, verifica-se a quantidade de chuva (em polegadas) e a quantidade de algodão colhido por acre em algumas estações do ano em um país produtor de algodão.

a) Faça um diagrama de pontos (scatter) que represente a quantidade chuva pela produção de algodão. Qual modelo ou função matemática mais apropriado para modelar esse problema?

b) Utilize o Excel para determinar essa função;

c) Com esse modelo determinado, estime qual será a produção de algodão para 25 polegadas de chuva.

Estações	Chuva (polegadas)	Produção (gk/acre)
1	20,3	5311
2	20,1	4382
3	18,1	3950
4	12,5	3137
5	30,9	5113
6	33,6	4814
7	35,8	3540
8	15,5	3850
9	27,6	5071
10	34,5	3881

```
# Dados fornecidos no problema
chuvas = np.array([20.3, 20.1, 18.1, 12.5, 30.9, 33.6, 35.8, 15.5, 27.6, 34.5])
producao = np.array([5311, 4382, 3950, 3137, 5113, 4814, 3540, 3850, 5071, 3881])

# Ajustando um modelo polinomial de segundo grau (quadrático)
coeficientes = np.polyfit(chuvas, producao, 2)
modelo = np.poly1d(coeficientes)

# Estimando a produção para 25 polegadas de chuva
estimativa_25_polegadas = modelo(25)

# Plotando o diagrama de dispersão e a curva do modelo
plt.figure(figsize=(10, 6))
plt.scatter(chuvas, producao, color='blue', label='Dados Originais')
# Gerando uma série de valores de chuva para plotar a curva do modelo
valores_chuva = np.linspace(min(chuvas), max(chuvas), 100)
plt.plot(valores_chuva, modelo(valores_chuva), 'r--', label='Curva do Modelo')
plt.title('Produção de Algodão por Chuva Recebida')
plt.xlabel('Chuva (polegadas)')
plt.ylabel('Produção (kg/acre)')
plt.legend()
plt.grid(True)
plt.show()

print(f"Estimativa da produção de algodão para 25 polegadas de chuva: {estimativa_25_polegadas:.2f} kg/acre")

```