

✓ Tratamento de Erros

```
1 # Parte 1 - Exercícios com Erros Comuns
2
3 # Exercício 1:
4
5 try:
6     def dividir(a, b):
7         return a / b # tratar ZeroDivisionError
8     num1 = 10
9     num2 = 0
10    resultado = dividir(num1, num2)
11    print('Resultado:', resultado)
12 except ZeroDivisionError:
13    print('Erro: Divisão por zero')
```



Erro: Divisão por zero

```
1  # Exercício 2:
2
3  try:
4      entrada = input('Digite um número inteiro: ')
5      numero = int(entrada) # tratar ValueError
6      print('Número:', numero)
7  except:
8      print('Entrada inválida. Certifique-se de digitar um número inteiro')
```



Digite um número inteiro: 5.5
Entrada inválida. Certifique-se de digitar um número inteiro

```
1 # Exercício 3:
2
3 lista = [1, 2, 3]
4 try:
5     indice = int(input('Índice a acessar: '))
6     print(lista[indice]) # tratar IndexError, ValueError
7 except IndexError:
8     print(f'Índice inválido: você digitou {indice}, o tamanho da lista é {len(lista)}')
9 except ValueError:
10    print('Entrada inválida. Certifique-se de digitar um número inteiro')
```



Índice a acessar: 5
Índice inválido: você digitou 5, o tamanho da lista é 3

```
1 # Exercício 4:
2
3 try:
4     arquivo = open('dados.txt', 'r') # tratar FileNotFoundError
5     conteudo = arquivo.read()
6     print(conteudo)
7     arquivo.close()
8 except FileNotFoundError:
9     print('Arquivo não encontrado')
```




Arquivo não encontrado

```
1 # Exercício 5:
2
3 valores = [10, 20, 'trinta', 40]
4 soma = 0
5
6 try:
7     for v in valores:
8         soma += v # tratar TypeError
9     print(soma)
10 except TypeError:
11    print('Erro: Não é possível somar um inteiro com uma string')
```




Erro: Não é possível somar um inteiro com uma string


```
1 # Exercício 6:
2
3 texto = 'Exemplo'
4
5 try:
6     print(texto.append('!')) # tratar AttributeError
7 except AttributeError:
8     print('Erro: Objeto string não possui o método append')
```

 Erro: Objeto string não possui o método append

```
1 # Exercício 7:
2
3 dados = {'nome': 'João'}
4
5 try:
6     print(dados['endereco']) # tratar KeyError
7 except KeyError:
8     print('Erro: Chave não encontrada no dicionário')
```

 Erro: Chave não encontrada no dicionário


```
1 # Exercício 8:
2
3 try:
4     print(resultado) # tratar NameError
5 except NameError:
6     print('Erro: Variável não definida')
```

 Erro: Variável não definida


```
1 # Exercício 9:
2
3 try:
4     import minha_biblioteca_falsa # tratar ModuleNotFoundError
5 except ModuleNotFoundError:
6     print('Erro: Biblioteca não encontrada')
```

 Erro: Biblioteca não encontrada

```
1 # Exercício 10:
2
3 numeros = [1, 2, 3]
4
5 try:
6     print(numeros[3]) # tratar IndexError
7 except IndexError:
8     print('Erro: Índice fora dos limites da lista')
```

 Erro: Índice fora dos limites da lista

```
1 # Parte 2 - Exercícios com Erros em Ciência de Dados
2
3 # Exercício 1:
4
5 # Criar um arquivo CSV vazio para teste
6 with open('arquivo_vazio.csv', 'w') as f:
7     f.write('') # linha vazia
8
9 import pandas as pd
10
11 try:
12     df = pd.read_csv('arquivo_vazio.csv') # tratar pandas.errors.EmptyDataError
13 except pd.errors.EmptyDataError:
14     print('Erro: Arquivo CSV vazio')
15 except FileNotFoundError:
16     print('Erro: Arquivo não encontrado')
```


 Erro: Arquivo CSV vazio

```
1 # Exercício 2:
2
3 # Criar um arquivo CSV mal formatado para teste
4 with open('arquivo_mal_formatado.csv', 'w') as f:
5     f.write('coluna1,coluna2\n')
6     f.write('valor1,valor2\n')
7     f.write('valor3, valor4, valor5\n') # Linha com mais colunas que o cabeçalho
```

```

8
9 import pandas as pd
10
11 try:
12     df = pd.read_csv('arquivo_mal_formatado.csv') # tratar pandas.errors.ParserError
13     print(df)
14 except pd.errors.ParserError:
15     print('Erro: Arquivo CSV mal formatado')
16 except FileNotFoundError:
17     print('Erro: Arquivo não encontrado')


```

 Erro: Arquivo CSV mal formatado

```

1 # Exercício 3:
2
3 import pandas as pd
4 import numpy as np
5
6 try:
7     df = pd.DataFrame({'A': [1, 1, 2, None], 'B': [3, 4, 5, 6]})
8     agrupado = df.groupby('X', dropna=False).sum() # tratar TypeError ou problemas com NaN
9     print(agrupado)
10 except Exception as e:
11     print(f'Erro: {e}')

```

 Erro: 'X'

```

1 # Exercício 4:
2
3 from sklearn.model_selection import train_test_split
4
5 try:
6     X = [1, 2, 3]
7     y = [0, 1]
8     train_test_split(X, y) # tratar ValueError
9 except ValueError:
10     print('Erro: O número de amostras em X e y deve ser o mesmo')


```

 Erro: O número de amostras em X e y deve ser o mesmo

```

1 # Exercício 5:
2
3 import pandas as pd
4
5 try:
6     df = pd.DataFrame({'A': [1, 2]})
7     print(df.loc[3]) # tratar KeyError ou IndexError
8 except KeyError:
9     print('Erro: Chave não encontrada no índice')
10 except IndexError:
11     print('Erro: Índice fora dos limites do DataFrame')


```

 Erro: Chave não encontrada no índice

```

1 # Exercício 6:
2
3 import numpy as np
4
5 try:
6     grande_array = np.ones((100000000, 100000000)) # tratar MemoryError
7 except MemoryError:
8     print('Erro: Não foi possível alocar a memória suficiente')


```

 Erro: Não foi possível alocar a memória suficiente

```

1 # Exercício 7:
2
3 import pandas as pd
4
5 df = pd.DataFrame({'A': [1, 2, 3]})
6 subset = df[df['A'] > 1].copy() # usado .copy() para evitar warning
7 subset['A'] = 10
8 print(subset)

```


 A

1	10
2	10

```

1 # Exercício 8:
2
3 import numpy as np
4 from numpy.linalg import inv
5
6 try:
7     matriz = np.array([[1, 2], [2, 4]])
8     inv(matriz) # tratar numpy.linalg.LinAlgError
9 except np.linalg.LinAlgError:
10    print('Erro: Matriz singular')

```

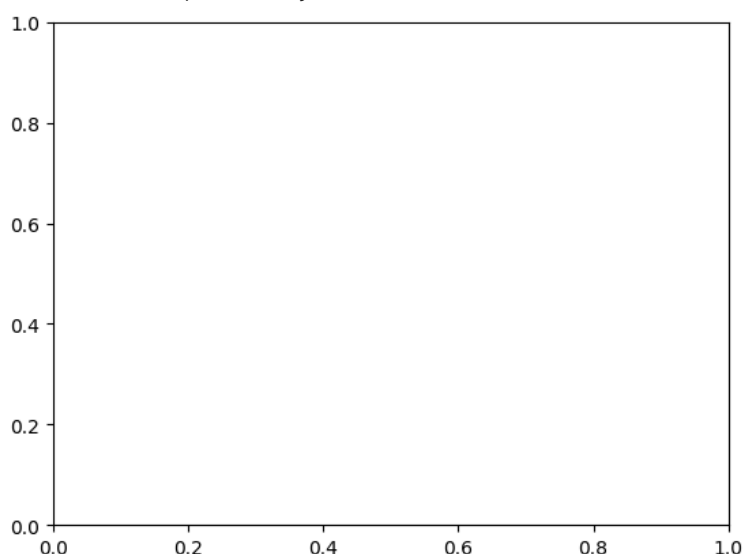
 Erro: Matriz singular

```

1 # Exercício 9:
2
3 import matplotlib.pyplot as plt
4
5 try:
6     x = [1, 2, 3]
7     y = [1, 2]
8     plt.plot(x, y) # tratar ValueError
9 except ValueError:
10    print('Erro: O número de pontos x e y deve ser o mesmo')

```


 Erro: O número de pontos x e y deve ser o mesmo



```

1 # Exercício 10:
2
3 # Criar um arquivo CSV vazio para teste
4 with open('arquivo.csv', 'w') as f:
5     f.write('coluna1') # linha vazia
6
7 import pandas as pd
8
9 try:
10    df = pd.read_csv('arquivo.csv')
11    df['nova'] = df['coluna_inexistente'] + 1 # tratar KeyError
12 except KeyError:
13    print('Erro: Chave não encontrada no DataFrame')

```

 Erro: Chave não encontrada no DataFrame

```

1 from scipy.stats import linregress
2
3 def fit_trendline(year_timestamp, data):
4     if not year_timestamp or not data:
5         raise ValueError("Timestamps e data não podem ser listas vazias.")
6
7     result = linregress (year_timestamp, data)
8     slope = round(result.slope, 3)
9     r_squared = round(result.rvalue**2, 3)
10    return slope, r_squared
11
12 timestamp = []
13 data = [18.36,18.36,17.91]
14 fit_trendline(timestamp, data)

```



```
ValueError                                Traceback (most recent call last)
<ipython-input-21-cc6b01a58b58> in <cell line: 0>()
    12 timestamp = []
    13 data = [18.36,18.36,17.91]
----> 14 fit_trendline(timestamp, data)

<ipython-input-21-cc6b01a58b58> in fit_trendline(year_timestamp, data)
     3 def fit_trendline(year_timestamp, data):
     4     if not year_timestamp or not data:
----> 5         raise ValueError("Timestamps e data não podem ser listas vazias.")
     6
     7     result = linregress (year_timestamp, data)

ValueError: Timestamps e data não podem ser listas vazias.
```

Próximas etapas: [Explicar o erro](#)

```
1  # Classe base de exceções
2  class NewException (Exception):
3      def __init__(self, message="Erro personalizado na aplicação."):
4          self.message = message
5          super().__init__(self.message)
6
7  # Exceção específica que herda de NewException
8  class InputInvalidoError (NewException):
9      def __init__(self, entrada, message="Entrada inválida fornecida."):
10         self.entrada = entrada
11         super().__init__(f"{message} Valor recebido: {entrada}")
12
13 # Uso em uma função
14 def processar_entrada (valor):
15     if not isinstance (valor, int):
16         raise InputInvalidoError(valor)
17     print(f"Valor processado com sucesso: {valor}")
18
19 # Testando com entrada incorreta
20 try:
21     processar_entrada ("abc")
22 except InputInvalidoError as e:
23     print("Erro capturado:", e)
```



Erro capturado: abc