1. def filtrar_impares(lista):

   # Criamos uma lista vazia para armazenar os números ímpares

   impares = []

   # Percorremos cada número da lista original

   for numero in lista:

      # Verificamos se o número é ímpar (ou seja, o resto da divisão por 2 é diferente de 0)

      if numero % 2 != 0:

         # Se for ímpar, adicionamos à nova lista

         impares.append(numero)

   # Retornamos a lista contendo apenas os números ímpares

   return impares

2. def eh_primo(numero):

Números menores que 2 não são primos

      if numero < 2:

            return False

Verificamos divisores de 2 até a raiz quadrada do número

    for i in range(2, int(numero ** 0.5) + 1):

        if numero % i == 0:

            return False

return True

def filtrar_primos(lista):

    primos = []

    for numero in lista:

        if eh_primo(numero):

```
            primos.append(numero)
    return primos
```

3.
```
def elementos_exclusivos(lista1, lista2):
        resultado = []
        for elemento in lista1:
            if elemento not in lista2 and elemento not in resultado:
                resultado.append(elemento)

        for elemento in lista2:
            if elemento not in lista1 and elemento not in resultado:
                resultado.append(elemento)

    return resultado
```

4.
```
def segundo_maior(lista):
    if len(lista) < 2:
        return None  # Ou lançar um erro, dependendo do caso

    maior = segundo = float('-inf')

    for numero in lista:
        if numero > maior:
            segundo = maior
            maior = numero
        elif maior > numero > segundo:
            segundo = numero

    # Se segundo continuar como -inf, é porque não havia segundo maior distinto
```

```
    return segundo if segundo != float('-inf') else None
```

5.
```
def ordenar_por_nome(lista_pessoas):
        return sorted(lista_pessoas, key=lambda pessoa: pessoa[0].lower())
pessoas = [("Carlos", 30), ("ana", 25), ("Beatriz", 22), ("davi", 28)]
ordenadas = ordenar_por_nome(pessoas)
print("Lista ordenada:")
for nome, idade in ordenadas:
    print(f"{nome} - {idade} anos")
```

6. Considera-se um outlier o valor que está muito distante da média, em sua maioria, a mais de três desvios padrão. Para resolver um outlier por desvio padrão, pode-se usar a fórmula: x < média - 3 * desvio padrão ou x > média + 3 * desvio padrão. Enquanto nos quartis, o outlier se dá por valore que estão fora de um intervalo considerado normal. Nesse caso, usa-se a fórmula: IQR = Q3 – Q1, onde o um valor x é outlier se: x < Q1 – 1.5 * IQR ou X > Q3 + 1.5 * IQR.

7. Por linhas:

```
import pandas as pd

# DataFrame 1
df1 = pd.DataFrame({
     'nome': ['Ana', 'Bruno'],
     'idade': [23, 30]
})

# DataFrame 2 com coluna diferente
df2 = pd.DataFrame({
      'nome': ['Carlos', 'Diana'],
```

```
    'altura': [1.75, 1.65]
})

# Concatenando pelas linhas
resultado = pd.concat([df1, df2], axis=0, ignore_index=True)
print(resultado)
```

Por colunas:

```
df1 = pd.DataFrame({
    'A': [1, 2],
    'B': [3, 4]
})

df2 = pd.DataFrame({
     'C': ['x', 'y']
})

# Concatenando pelas colunas
resultado = pd.concat([df1, df2], axis=1)
print(resultado)
```

8.

```
import pandas as pd
# Lê o arquivo CSV (substitua 'caminho/arquivo.csv' pelo caminho real do seu arquivo)
df = pd.read_csv('caminho/arquivo.csv')
# Exibe as 5 primeiras linhas do DataFrame
```

```
print(df.head())
```

9. import pandas as pd

```
df = pd.read_csv('dados.csv')  # Exemplo de leitura de um CSV

# Selecionando a coluna "idade"
coluna_idade = df['idade']
print(coluna_idade)
filtro = df[df['idade'] > 30]
print(filtro)
```

10. import pandas as pd

    import numpy as np

```
dados = {
    'nome': ['Ana', 'Bruno', 'Carlos', 'Diana'],
    'idade': [25, np.nan, 30, 22],
    'altura': [1.65, 1.80, np.nan, 1.55]
}

df = pd.DataFrame(dados)

print("Detectar NaNs:")
print(df.isna())
```

```
print("\nRemover linhas com NaN:")

print(df.dropna())


print("\nPreencher NaNs na coluna idade com a média:")

df['idade'] = df['idade'].fillna(df['idade'].mean())

print(df)
```