**Códigos desafio akaer**

1. **Python**

def encontrar\_faltante(n, numeros):

# Calcula a soma esperada com a fórmula da soma de uma progressão aritmética

soma\_esperada = (n \* (n + 1)) // 2

# Calcula a soma real dos números fornecidos

soma\_real = sum(numeros)

# O número faltante é a diferença entre a soma esperada e a soma real

numero\_faltante = soma\_esperada - soma\_real

return numero\_faltante

# Entrada de exemplo

n = int(input("Digite o valor de N: "))

numeros = list(map(int, input("Digite a lista de N-1 inteiros separados por espaço: ").split()))

# Chama a função e imprime o resultado

resultado = encontrar\_faltante(n, numeros)

print(f"O número faltante na sequência é: {resultado}")

1. **- Python**

def contar\_pares\_corretos(lista\_botas):

contagem\_botas = {}

pares\_corretos = 0

for bota in lista\_botas:

tamanho, lado = bota

if tamanho in contagem\_botas:

if contagem\_botas[tamanho] != lado:

pares\_corretos += 1

del contagem\_botas[tamanho]

else:

contagem\_botas[tamanho] = 'D' if lado == 'E' else 'E'

else:

contagem\_botas[tamanho] = lado

return pares\_corretos

n = int(input())

lista\_botas = []

for \_ in range(n):

bota = tuple(input().split())

lista\_botas.append(bota)

resultado = contar\_pares\_corretos(lista\_botas)

print(resultado)

1. **Python**

import pandas as pd

# Ler o arquivo de entrada

df = pd.read\_excel('Input\_Teste\_Python\_exercicio 3.xlsx')

# Converte as colunas de hora para o formato datetime

df['Start Time'] = pd.to\_datetime(df['Start Time'])

df['End Time'] = pd.to\_datetime(df['End Time'])

# Calcula a diferença entre as horas e converte para hora:minuto

df['Tempo Resultante'] = (df['End Time'] - df['Start Time']).dt.total\_seconds() / 60

# Extrai o dia (dia/mes/ano) da coluna 'Hora Inicial'

df['Dia'] = df['Start Time'].dt.strftime('%d/%m/%Y')

# Converte os minutos para o formato hora:minuto

df['Tempo Resultante'] = pd.to\_timedelta(df['Tempo Resultante'], unit='m')

df['Tempo Resultante'] = df['Tempo Resultante'].astype(str).str.slice(7, -3)

# Seleciona as colunas desejadas na tabela final

tabela\_final = df[['User Name', 'License Type', 'Dia', 'Tempo Resultante']]

# Imprime a tabela final

print(tabela\_final)

**4) Python**

import pandas as pd

from tabulate import tabulate

def encontrar\_relacoes(df\_base, df\_licencas):

df\_base['Username'] = df\_base.apply(lambda row\_base:

df\_licencas.loc[

(df\_licencas['Hora Inicial'] <= row\_base['Hora Inicial']) &

(row\_base['Hora Inicial'] <= df\_licencas['Hora Final']),

'Usuario'

].iloc[0] if not df\_licencas[

(df\_licencas['Hora Inicial'] <= row\_base['Hora Inicial']) &

(row\_base['Hora Inicial'] <= df\_licencas['Hora Final'])

].empty else 'not found', axis=1)

return df\_base

# Leitura dos dados dos arquivos externos

df\_licencas\_especiais = pd.read\_excel('licencas\_especiais.xlsx')

df\_base\_comum = pd.read\_excel('base\_comum.xlsx')

# Converta as colunas de hora para datetime

df\_licencas\_especiais[['Hora Inicial', 'Hora Final']] = df\_licencas\_especiais[['Hora Inicial', 'Hora Final']].apply(pd.to\_datetime)

df\_base\_comum[['Hora Inicial', 'Hora Final']] = df\_base\_comum[['Hora Inicial', 'Hora Final']].apply(pd.to\_datetime)

# Encontrar relações

df\_base\_comum = encontrar\_relacoes(df\_base\_comum, df\_licencas\_especiais)

# Exiba as relações encontradas e não encontradas

encontrados = df\_base\_comum[df\_base\_comum['Username'] != 'not found']

nao\_encontrados = df\_base\_comum[df\_base\_comum['Username'] == 'not found']

# Imprima as tabelas

print("Relações encontradas:")

print(tabulate(encontrados[['ID\_Usuario', 'Username', 'Licenca', 'Hora Inicial', 'Hora Final']], headers='keys', tablefmt='fancy\_grid'))

print("\nRelações não encontradas:")

print(tabulate(nao\_encontrados[['ID\_Usuario', 'Licenca', 'Hora Inicial', 'Hora Final']], headers='keys', tablefmt='fancy\_grid'))

1. **SQL**

SELECT

c.name,

o.id AS order\_id

FROM

Customers c

JOIN

Orders o ON c.id = o.id\_customers

WHERE

EXTRACT(MONTH FROM o.orders\_date) BETWEEN 1 AND 6

AND EXTRACT(YEAR FROM o.orders\_date) = 2016;

**2)**

SELECT

p.name AS product\_name,

c.name AS category\_name

FROM

products p

JOIN

categories c ON p.id\_categories = c.id

WHERE

p.amount > 100

AND p.id\_categories IN (1, 2, 3, 6, 9)

ORDER BY

product\_name, category\_name;

**3)**

SELECT c.name AS category\_name, SUM(p.amount) AS total\_amount

FROM categories c

JOIN products p ON c.id = p.id\_categories

GROUP BY c.name;