A. Revisão rápida

Conteúdo: Sintaxe básica, listas, dicionários, laços, funções.

1. Organização de nomes

Crie uma função chamada organizar_nomes(lista_nomes) que:

- Recebe uma lista de nomes (com duplicatas e maiúsculas/minúsculas misturadas).
- Remove duplicatas.
- Retorna os nomes **ordenados em ordem alfabética** e **padronizados com a primeira letra maiúscula**.

Exemplo:

```
entrada = ["ana", "Carlos", "joão", "ana", "MARIA", "carlos"]
saida = organizar_nomes(entrada)
print(saida)
# ['Ana', 'Carlos', 'João', 'Maria']
```

2. Contagem de palavras em um texto

Escreva uma função *contar_palavras(texto)* que:

- Recebe uma string.
- Conta quantas vezes cada palavra aparece.
- Retorna um **dicionário** com a contagem.

Exemplo:

```
texto = "Python é divertido e Python é poderoso"
print(contar_palavras(texto))
# {'Python': 2, 'é': 2, 'divertido': 1, 'e': 1, 'poderoso': 1}
```

(dica: usar split() e laço for para atualizar o dicionário)

3. Filtrando números primos

Crie uma função filtrar_primos(lista_numeros) que:

- Recebe uma lista de números inteiros.
- Retorna apenas os **números primos** em uma nova lista.

Exemplo:

```
numeros = [2, 3, 4, 5, 9, 11, 15, 17]
print(filtrar_primos(numeros))
# [2, 3, 5, 11, 17]
```

B. Python moderno e produtivo

Conteúdo:

- List comprehensions e dict comprehensions.
- Manipulação de strings e expressões regulares.
- Manipulação de arquivos (texto, JSON, CSV).
- Uso de bibliotecas essenciais: numpy: arrays, operações vetorizadas; pandas: DataFrames, seleção, filtragem, agregação.

Mini-desafio: ler um CSV simples, processar dados e gerar estatísticas

1. List & Dict Comprehensions

- Crie uma lista de números de 1 a 20.
- Usando **list comprehension**, gere uma nova lista apenas com os **quadrados dos números pares**.
- Usando **dict comprehension**, crie um dicionário em que a chave seja o número e o valor seja "par" ou "ímpar".

Saída esperada (resumida):

```
quadrados_pares = [4, 16, 36, ..., 400]
classificacao = {1: 'ímpar', 2: 'par', 3: 'ímpar', ...}
```

2. Manipulação de Strings e Regex

Escreva uma função extrair_emails(texto) que:

- Recebe o endereço de um arquivo contendo um texto (Arquivo exemplo: ia_1_exemplo_ata_reuniao.txt).
- Usa **expressões regulares** para encontrar todos os endereços de e-mail válidos.
- Retorna a lista de e-mails encontrados.

Exemplo de saída:

```
print(extrair_emails(texto))
# ['maria@gmail.com', 'joao@yahoo.com']
```

3. Manipulação de Arquivos (JSON)

A partir do uso da API Nomes do IBGE (https://servicodados.ibge.gov.br/api/docs/nomes?versao=2), construa um programa que, dado uma lista de nomes próprios como entrada, realize consultas do tipo “Frequência por nome” à API, e apresente o resultado em tela no seguinte formato:

```
{
‘Abner’: 35485,
‘Ana’: 451884,
```

‘Davi’: 54848,
‘Samuel’: 45378‘
}

4. NumPy – Operações Vetorizadas

Com **NumPy**, crie um array 1D com 10 números aleatórios entre 1 e 100.

- Calcule a **média**, o **máximo** e o **mínimo**.
- Crie um novo array apenas com os números **maiores que a média**.
- Normalize o array (valores entre 0 e 1).

Use funções numpy e operações vetorizadas.

5. Pandas – Manipulação de dados

Dado o arquivo ia1_vendas_carros.csv:

- Leia o CSV com **pandas**.
- Realize operações de pré-processamento, caso necessário.
- Calcule:
    1. Valor médio por modelo.
    2. Valor médio por ano do modelo.
    3. Frequência de vendas por modelo.
    4. Maior preço por modelo.
    5. Menor preço por modelo

C. Revisão de Machine Learning Básico com Scikit-learn

Regressão: prever preços de veículos usando regressão linear e polinomial.Use a base de dados ia1_vendas_carros.csv.