Aula 10

Estruturas de Dados - Parte 2
(Tuplas e Dicionários)

Algoritmos e Estrutura de Dados I

1º Semestre - CDN

Prof. Dr. Dilermando Piva Jr.

Conteúdo Programático - Planejamento

Conteúdo Programático		
Semana	**Data**	**Temas/Atividades**
1	**12/08**	Acolhimento e Boas-vindas! Introdução a Disciplina. Formas de Avaliação e Percurso Pedagógico.
2	**19/08**	Pensamento Computacional. O que é qual sua importância para Ciência de Dados
3	**26/08**	Primeiro Programa – Variáveis, Tipos de Dados e Saída em Python
4	**30/08**	*/reposição/ Introdução a Computação e representação da informação –História e evolução da computação. A informação e sua representação. Conversão entre bases.*
5	**02/09**	Operadores, Cálculo Simples e Entrada de Dados. Estruturas sequenciais.
6	**09/09**	Tomando Decisões: Estrutura Condicional (if/else)
7	**16/09**	Repetição de Ações: Estrutura de Repetição – Introdução aos loops (for e while)
8	**23/09**	Prática Integrada – Construindo um Jogo Simples (Adivinhação) – Pj1
9	**30/09**	Estruturas de Dados Parte 1 – listas e sequências (e strings)
10	**07/10**	Estruturas de Dados Parte 2 – Dicionários e Dados Estruturados
11	**14/10**	Integração – Projeto de Análise de Dados Simples com Listas e Dicionários – Pj2
12	**21/10**	***Primeira Avaliação Formal. (P1).*** Correção da Avaliação após o intervalo.
13	**28/10**	Modularização do código – Introdução a Funções
14	**04/11**	Parâmetros, Retornos e Boas Práticas de Funções
15	**11/11**	Continuação de funções, Módulos e Pacotes em Python
16	**18/11**	Introdução a Machine Learning – Pj3
17	**25/11**	***Semana de Apresentação PI de CDN***
18	***02/12***	Introdução a Ciência de Dados – partes 1 e 2 – Pandas, Numpy, Pipleline e exemplos – Pj4
19	***09/12***	***Segunda Avaliação Formal (P2).*** Correção da Avaliação após o intervalo
20	**16/12**	***Exame / Avaliação Substitutiva.*** Divulgação do Resultado Final.

Tuplas

Tuplas...

- São coleções (caracteres, strings, números...) ordenados, separados por vírgula e que estão dentro de parenteses (limitados por parenteses).
- São coleções IMUTÁVEIS

- *Exemplo de Tuplas Homogêneas:*

(1,2,3,4,...)

(‘a’, ‘b’, ‘c’, ...)

(‘texto1’, ‘texto2’, ...)

- *Exemplo de Tuplas Heterogêneas:*

(1, ‘a’, ‘texto1’, (1, 2, 3), 345, ...)

Tuplas... Exemplos práticos...

- *Exemplo 1 (cotidiano):* **Coordenadas geográficas:** latitude e longitude de um local. Por exemplo, coordenadas_sp = (-23.55, -46.63) representando São Paulo.
- *Exemplo 2 (negócios):* **Registro de produto imutável:** suponha que cada produto tenha um código e um nome que não mudam – por exemplo, código interno e nome de um item de estoque. Podemos armazenar isso em uma tupla, como produto = (101, "Cadeira Executiva").
- *Exemplo 3:* Outros exemplos rápidos podem ser mencionados: dias da semana como tupla (dias = ("Seg", "Ter", ..., "Dom") – uma coleção fixa de constantes).

Esses casos ilustram quando utilizar tuplas: **dados constantes ou agrupamentos lógicos de itens diferentes que não precisam ser alterados após definidos**.

Tuplas...

- **Atribuição e tipo**
- Operações básicas
- A função tuple()
- Indexação e Fatiamento de Tuplas

Tuplas...

```
///// ATRIBUIÇÃO E TIPO //////
# Criando uma Tupla
# nome_tupla = (item1, item2,  ...  , itemn)

montadoras = ("Ferrari", "Fiat", "Mercedez", "Renault")

montadoras
("Ferrari", "Fiat", "Mercedez", "Renault")

type(montadoras)
<class 'tuple'>

# Pode-se criar uma tupla vazia.
tupla_vazia = ()

# As tuplas podem ser heterogêneas
tupla_mista = (2022, 34.56, montadoras, "pyPRO")

tupla_mista
(2022, 34.56, ('Ferrari', 'Fiat', Mercedez', 'Renault'), 'pyPRO')
```

Tuplas...

- Atribuição e tipo
- **Operações básicas**
- A função tuple()
- Indexação e Fatiamento de Tuplas

Tuplas...

```
//// OPERAÇÕES BÁSICAS   //////
tupla1 = (2, 0, 1, 4)
tupla2 = (2, 0, 1, 9)
tupla1 + tupla2
tupla1*3
tupla1 == tupla2

2 in tupla1
True
8 in tupla1
False

tupla2 > tupla1
True
```

Tuplas...

- Atribuição e tipo
- Operações básicas
- **A função tuple()**
- Indexação e Fatiamento de Tuplas

Tuplas...

```
///// Função tuple()
navegadores = "vikings"
texto_para_tupla = tuple(navegadores)
texto_para_tupla
('v','i','k','i','n','g','s')
lista = [23, 34, "Nome", "A", 3.45]
lista_para_tupla = tuple(lista)
(23, 34, 'Nome', 'A', 3.45)
letras = ('a', 'b', 'c')
numeros = (1, 2, 3)
tuplas_aninhadas = (letras, numeros)
tuplas_aninhadas
(('a', 'b', 'c'),(1, 2, 3))
```

Tuplas...

- Atribuição e tipo
- Operações básicas
- A função tuple()
- **Indexação e Fatiamento de Tuplas**

Tuplas...

```
//// INDEXAÇÃO E FATIAMENTO DE TUPLAS
tupla1
(2, 0, 1, 4)
tupla1[1]
2
tupla1[-1]
4
#Fatiamento  ⟶ mesmo que as listas
cores = ('r', 'g', 'b', 'y', 'o', 'm')
cores
('r', 'g', 'b', 'y', 'o', 'm')
cores[1:4]
('g', 'b', 'y')
```

Tuplas...

- Funções builtins usados em tuplas

Função	Descrição
`len()`	Retorna o número de elementos de uma tupla
`sum()`	Retorna a soma dos números de uma tupla
`sorted()`	Retorna uma cópia modificada (ordenada) de uma tupla e mantem a tupla original intacta.

Tuplas...

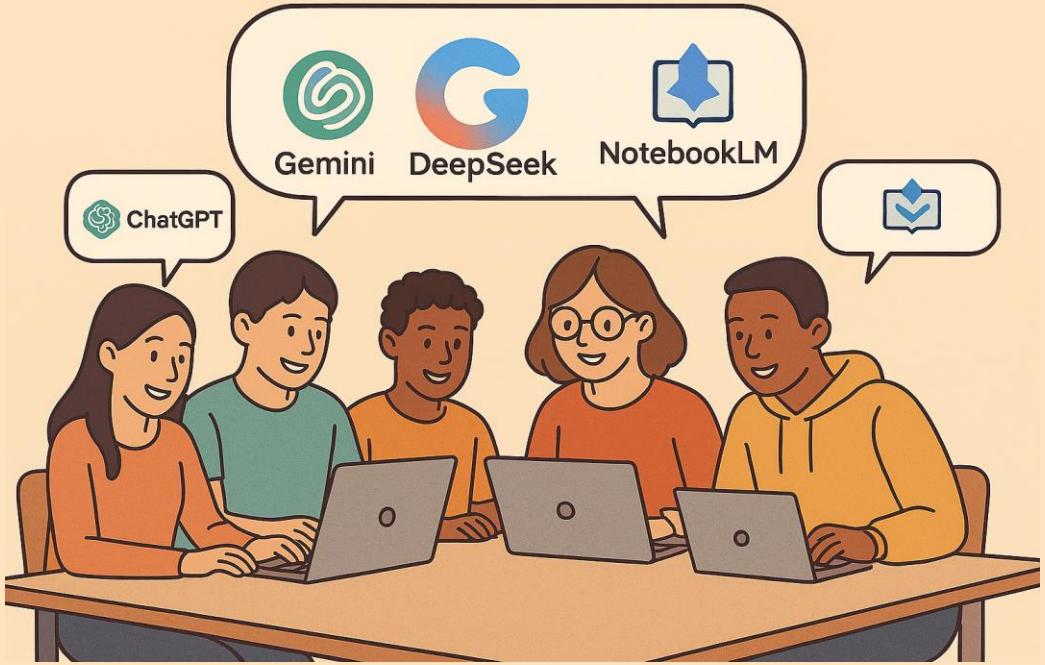
- Métodos de uma tupla.

Função	Sintaxe	Descrição
`count()`	`Tupla.count(item)`	Retorna a quantidade de vezes que um item ocorre em uma tupla
`index()`	`Tupla.index(item)`	Busca um item em uma tupla e retorna a sua posição. Se tiver mais de uma ocorrência do item, retorna a primeira. Se não encontrar o item, retorna Erro para o método.

"Tupla" deve ser substituído pelo nome atual da tupla (nome da variável)

Atividade com IA

- Vamos saber mais sobre:
  - Métodos do tipo **tuple**



- Faça individualmente, e depois compartilhe com o seu colega esses conceitos.
  - Por que a tupla tem menos métodos que a lista? O que isso revela sobre sua natureza?
  - Qual é a diferença entre as funções index() e count() em uma tupla?
  - Como posso acessar elementos de uma tupla e o que acontece se eu tentar modificá-los?
  - Ao trabalhar com tuplas, percebi que não consigo usar métodos como append() e remove(). Isso é uma limitação ou uma vantagem?
  - Quando devo preferir uma tupla em vez de uma lista? Pode me dar exemplos práticos com explicação?
  - Qual é a diferença entre tuple() e declarar uma tupla com parênteses?
  - Quais são os impactos de usar tuplas em termos de performance e segurança em sistemas maiores?

Vamos praticar...

Fazer um programa em python que:

- Receba uma quantidade ilimitada (não se sabe quantas) de idade de pessoas.
- Guarde essas idades em uma lista.
- Para encerrar a digitação das idades, deve-se digitar: -1
- Em seguida, deve-se gerar uma tupla de idades, a partir da lista.
- Deve-se mostrar as seguintes informações, consultando a tupla:
  - Qtd de idades digitadas
  - Média das idades

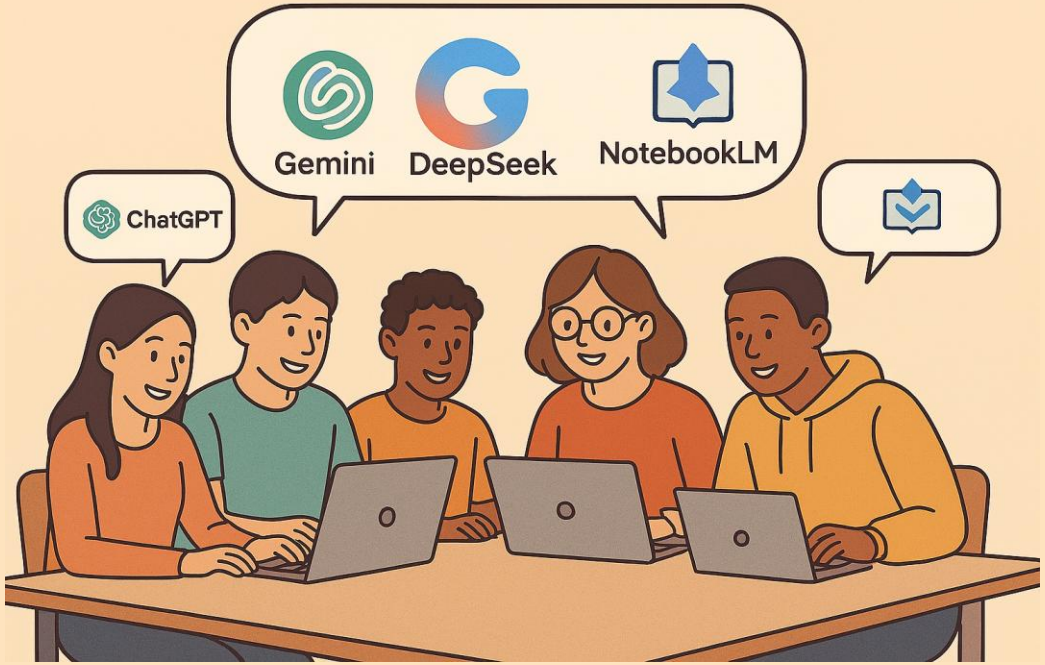
Possível solução...

```python
lista_idades = []
while True:
    idade = int(input("Entre com uma idade: "))
    if idade != -1:
        lista_idades.append(idade)
    else:
        break

tupla_idades = tuple(lista_idades)
total = sum(tupla_idades)
qtd = len(tupla_idades)
media = total / qtd
print(f"Total de idades digitadas: {qtd}")
print(f"A média de idades é: {media}")
```

Atividade com IA

- Vamos saber mais sobre:
  - Utilização de tuplas...



- Faça individualmente, e depois compartilhe com o seu colega esses conceitos.
  - “Pode me explicar, detalhadamente como funciona o seguinte código?” coloque o código do slide anterior.
  - O que a IA falou de diferente? Como você traduziria em outras palavras a explicação do código?
  - Existe algum outro método para resolver o problema utilizando tuplas?

Dicionários

Dicionários...

- São coleções de um conjunto de pares chave:valor, com uma restrição de que a chave seja única dentro da coleção.
- Os dicionários são construídos, utilizando chaves {}
- Utiliza-se “:” para separar a chave do valor
- Sempre o operador a esquerda é a chave e o a direita, o valor.

Os dicionários são indexados pela chave.

São as chaves“case sensitive”

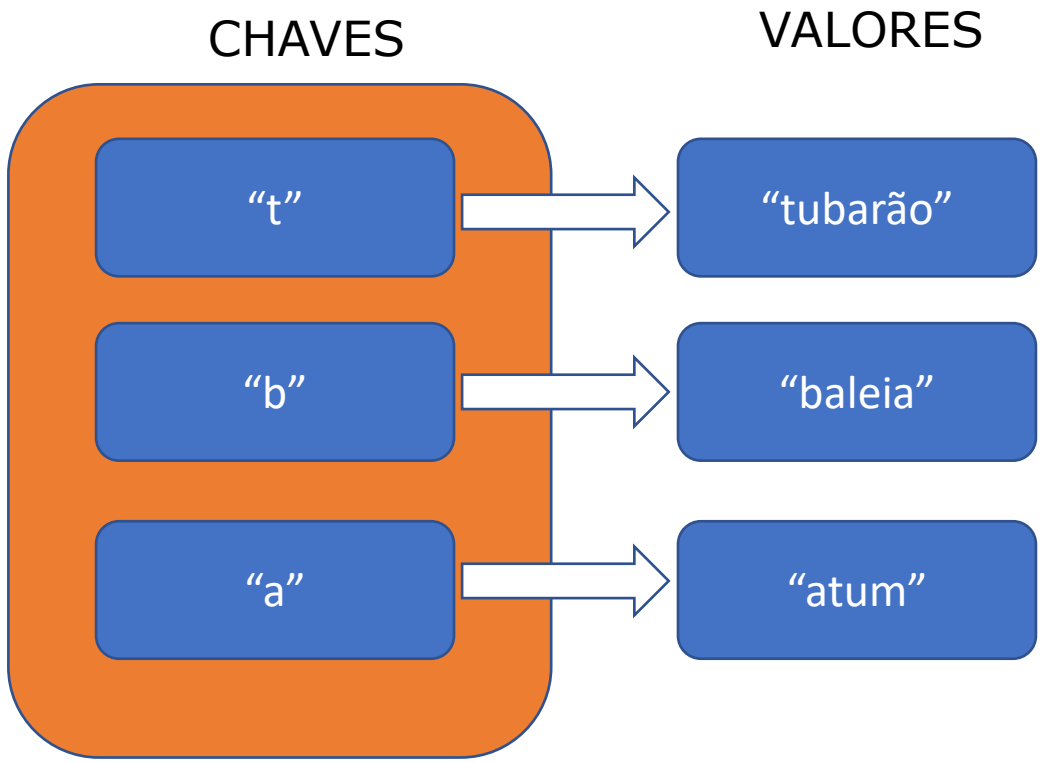
- *Exemplo de Dicionários:*

peixes = {“t”:”tubarão”, “b”:”baleia”, “a”:”atum”}

f1 = {1:”Mercedez”, 2:”Ferrari”, 3:”Renault”}

Dicionários...

peixes = {“t”:”tubarão”, “b”:”baleia”, “a”:”atum”}



Dicionários...

- *Exemplo de Dicionários (heterogêneo):*

dic_misto = {1:45, ‘a’:34.56, ‘teste’:’frase’, 3:’pyPRO’}

Dicionários... Exemplos práticos...

- *Exemplo 1 (negócios):* **Tabela de preços de produtos.** chaves são nomes (ou códigos) de produtos e os valores são preços. Por exemplo: tabela_precos = {"Batata": 1.20, "Tomate": 2.30, "Alface": 0.45} representando preço de itens em Reais.
- *Exemplo 2 (cotidiano):* **Agenda telefônica simples.** chaves são nomes de pessoas e os valores são seus telefones: contatos = {"Ana": "11987654321", "Bruno": "11881234765"}.
- *Exemplo 3 (dados estruturados):* **Perfil de cliente.** os valores de um dicionário podem ser de qualquer tipo, inclusive listas ou tuplas. Por exemplo: cliente = {"id": 101, "nome": "Alice", "compras": [250.00, 125.50, 89.90]} onde as chaves identificam atributos de um cliente e os valores são o identificador, o nome e a lista de valores de compras recentes.
- *Exemplo 4:* **Conversão de moedas (mapa de códigos para nomes):** um pequeno dicionário moedas = {"USD": "Dólar Americano", "EUR": "Euro", "BRL": "Real Brasileiro"}.

Dicionários...

- **Atribuição e tipo**
- Operações básicas
- A função dict()
- Indexação e Alteração

Dicionários...

```
///// ATRIBUIÇÃO E TIPO //////
# Criando um Dicionário

peixes = {"t":"tubarão", "b":"baleia", "a":"atum"}
f1 = {1:"Mercedez", 2:"Ferrari", 3:"Renault"}

type(f1)
<class 'dict'>

# Pode-se criar um dicionário vazio.
dic_vazio = {}

# Os dicionários podem ser heterogêneas
dic_misto = {1:45, 'a':34.56, 'teste':'frase', 3:'pyPRO'}
```

Dicionários...

- Atribuição e tipo
- **Operações básicas**
- A função dict()
- Indexação e Alteração

Dicionários...

```
//// OPERAÇÕES BÁSICAS   //////
peixes = {"t":"tubarão", "b":"baleia", "a":"atum"}
f1 = {1:"Mercedez", 2:"Ferrari", 3:"Renault"}

peixes == f1
```

Dicionários...

- Atribuição e tipo
- Operações básicas
- **A função dict()**
- Indexação e Alteração

Dicionários...

```
///// Função dict()

/// dict([**kwarg])
numeros = dict(um=1, dois=2, tres=3)
numeros
{'um':1, 'dois':2, 'tres':3}

/// dict(iterável[, **kwarg])
dic1 = dict([('ana', 342), ('pedro', 3453), ('marcio',8987)])
dic1
{'ana':342, 'pedro':3453, 'marcio':8987}
```

Dicionários...

- Atribuição e tipo
- Operações básicas
- A função dict()
- **Indexação e Alteração**

Dicionários...

```
//// INDEXAÇÃO e ALTERAÇÃO
dic1
{'ana':342, 'pedro':3453, 'marcio':8987}

dic1['ana']
342

dic1['piva'] = 3456

dic1
{'ana':342, 'pedro':3453, 'marcio':8987, 'piva':3456}

dic1['jose']
<<erro>>
```

Dicionários...

- Funções builtins usados em dicionários

Função	Descrição
`len()`	Retorna o número de duplas (chave:valor) de um dicionário
`all()`	Retorna TRUE se todos os valores das chaves do dicionário forem verdadeiras. Caso contrário, FALSE.
`any()`	Retorna TRUE se pelo menos um valor das chaves do dicionário for verdadeiro. Caso contrário, FALSE.
`sorted()`	Retorna uma **lista** modificada (ordenada) de um dicionário baseado em suas chaves. Mantem intacto o dicionário.

Dicionários...

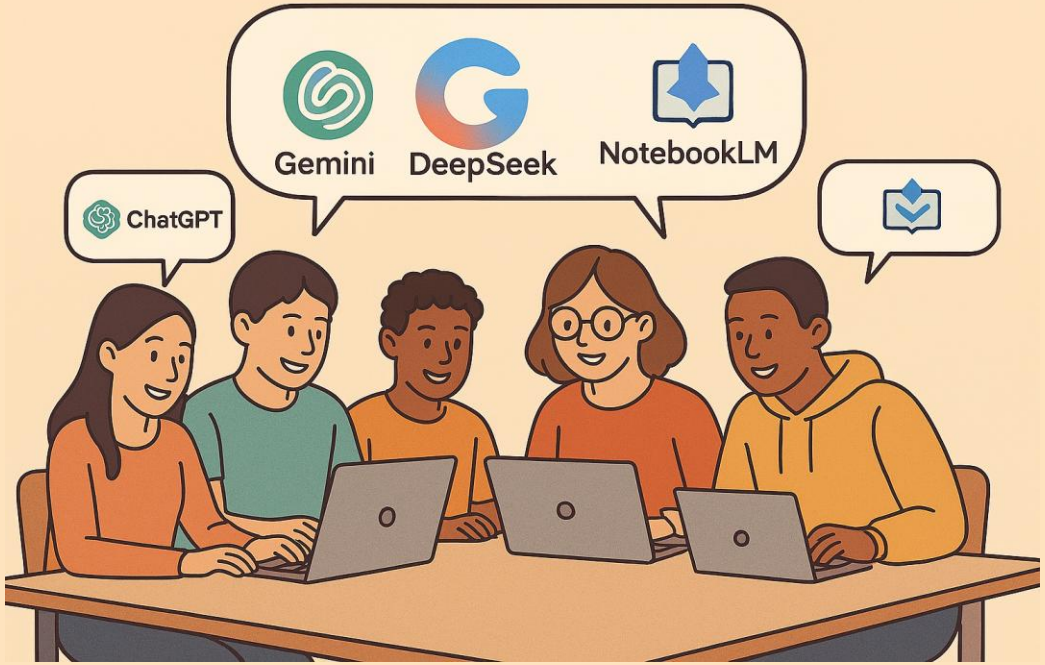
- Métodos de um dicionário.

Função	Sintaxe	Descrição
`clear()`	`Dic.clear()`	Remove todos os pares chave:valor de um dicionário.
`fromkeys()`	`Dic.fromkeys(seq[,valor])`	Cria um novo dicionário a partir da sequencia dos elementos com um valor informado
`get()`	`Dic.get(chave[,padrão])`	Retorna o valor associado à chave especificada no dicionário. Se a chave não estiver presente, ela retornará o valor padrão. Se padrão não for fornecido, o padrão será *None*, para que esse método nunca gere um KeyError.
`items()`	`Dic.items()`	Retorna uma nova visão dos pares de chave e valor do dicionário como tuplas.
`keys()`	`Dic.keys()`	Retorna uma nova view que consiste em todas as chaves do dicionário.
`pop()`	`Dic.pop(chave[,padrão])`	Remove a chave do dicionário e retorna seu valor. Se a chave não estiver presente, ela retornará o valor padrão. Se o padrão não for fornecido e a chave não estiver no dicionário, isso resultará em KeyError.
`popitem()`	`Dic.popitem()`	Remove e retorna um par de tuplas arbitrário (chave, valor) do dicionário. Se o dicionário estiver vazio, chamar popitem() resultará em KeyError.
`setdefault()`	`Dic.setdefault(chave[,padrão])`	Retorna um valor para a chave presente no dicionário. Se a chave não estiver presente, insere a chave no dicionário com um valor padrão e retorna o valor padrão. Se a chave estiver presente, o padrão é None, para que esse método nunca gere um KeyError.
`update()`	`Dic.update([outro])`	Atualiza o dicionário com os pares chave:valor de outro objeto de dicionário e retorna None.
`values()`	`Dic.values()`	Retorna uma nova view que consiste em todos os valores do dicionário.

"Dic" deve ser substituído pelo nome atual do dicionário (nome da variável)

Atividade com IA

- Vamos saber mais sobre:
  - Métodos do tipo **dict**



- Faça individualmente, e depois compartilhe com o seu colega esses conceitos.
  - Quais são os principais métodos disponíveis para objetos do tipo dict em Python e para que serve cada um?
  - Qual a diferença entre get() e o acesso direto usando colchetes ([]) em dicionários?
  - O que o método items() retorna e como posso usá-lo em um loop?
  - Para que serve o método update() e em que casos ele é útil?
  - O método pop() remove um item do dicionário. O que acontece se a chave não existir?
  - O que são dict_keys, dict_values e dict_items e por que esses objetos não são listas?
  - Compare os métodos pop(), del e clear(). Em que contexto cada um é mais adequado?
  - É possível usar setdefault() em vez de uma verificação com if chave in dicionario? Quando é melhor?
  - Qual a diferença prática entre dict.get() e usar try/except para capturar um KeyError? Quando cada abordagem é mais recomendada?

Vamos praticar...

Fazer um programa em python que:

- Receba o cadastro de uma qtd ilimitada (não se sabe quantas) de proprietários e seus respectivos apartamentos (número do apto e nome do proprietário)
- Guarde essas informações em um dicionário, onde a chave de busca é o número do apto.
- Para encerrar a digitação o número do apto será -1
- Em seguida, deve-se mostrar uma listagem em ordem crescente de apto: apto – nome do proprietário
- No final apresente a quantidade total de aptos listados.

Uma possível resposta...

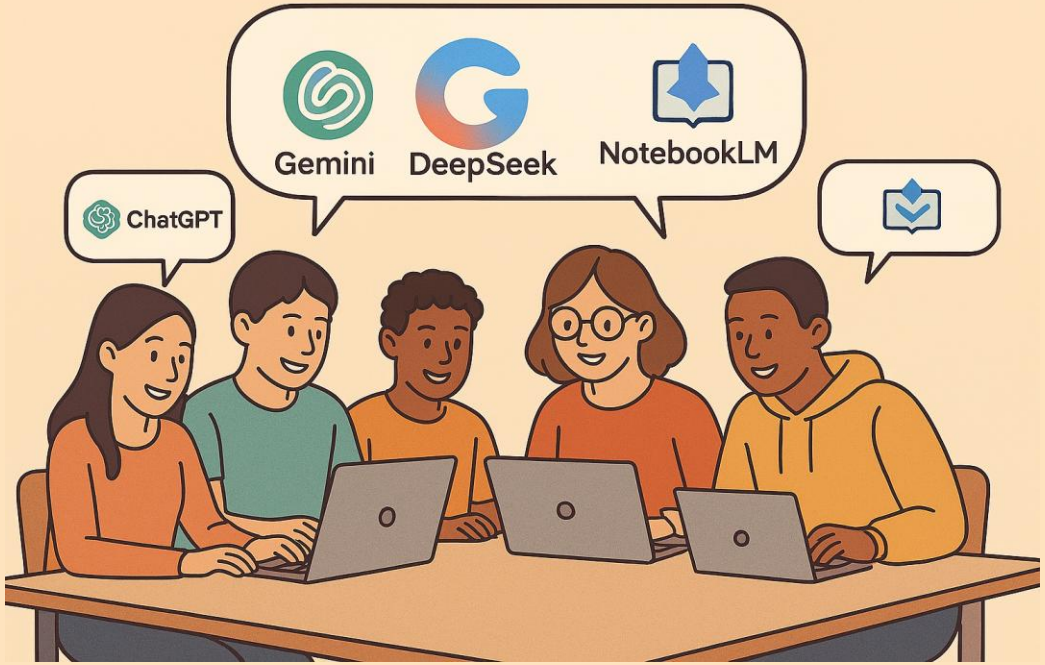
```
proprietarios = {}
while True:
    apto = int(input("Digite o apto: "))
    if apto != -1:
        proprietario = input("Proprietário: ")
        proprietarios.update({apto: proprietario})
    else:
        break

edificio = dict(sorted(proprietarios.items()))

for chave, valor in edificio.items():
    print(f"{chave} - {valor}")
```

Atividade com IA

- Vamos saber mais sobre:
  - Utilização de dicionários...



- Faça individualmente, e depois compartilhe com o seu colega esses conceitos.
  - “Pode me explicar, detalhadamente como funciona o seguinte código?” coloque o código do slide anterior.
  - O que a IA falou de diferente? Como você traduziria em outras palavras a explicação do código?
  - Existe algum outro método para resolver o problema utilizando dicionários?

VAMOS PARA A PRÁTICA ?!!!

Desafio 1

Contagem de frequência de palavras:

Receba uma frase do usuário. Pegue essa frase (string) e faça a contagem da ocorrência de cada palavra ou letra.

Exemplo: frase = "data science para negocios e ciencia de dados para negocios".

Converter para lista de palavras e usar dicionário para contar cada palavra.

Resultado seria algo como {"para": 2, "negocios": 2, "data": 1, "science": 1, "e": 1, "ciencia": 1, "dados": 1}.

Isso enfatiza que dicionário é excelente para análises de frequência (palavras em texto, produtos vendidos, etc.). É um exemplo cotidiano que pode ser expandido em data science (contar palavras-chave em avaliações de clientes, por exemplo). Aproveitar para relembrar string -> lista (usando .split() talvez, se já visto na semana de strings).

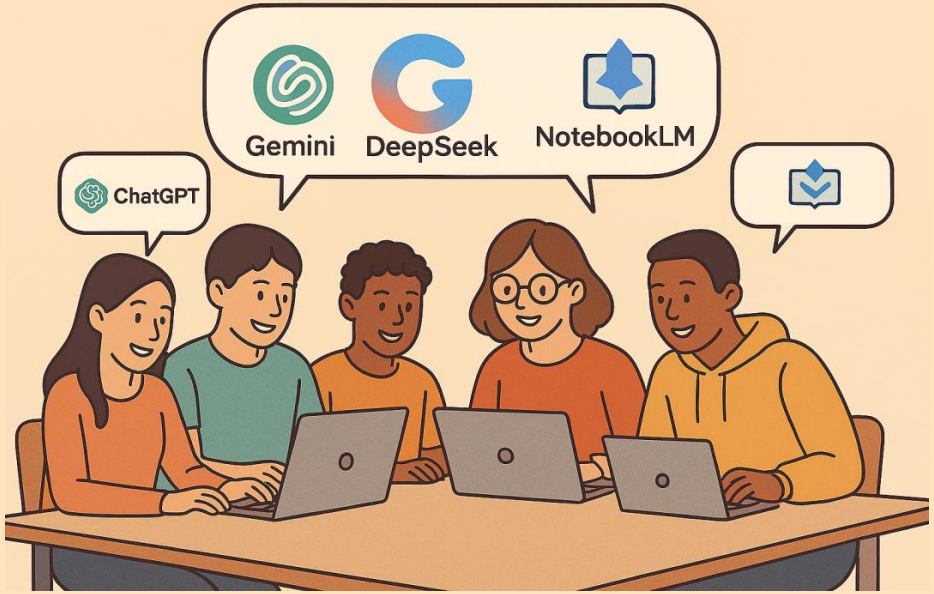
Desafio 2

Análise de Vendas: *Contexto:* Uma loja de e-commerce quer um relatório simples do desempenho de vendas.
Enunciado: Você recebe: (a) uma lista com os valores de vendas realizadas numa semana (lista de números float, representando, por exemplo, o faturamento de cada pedido); (b) uma lista com os nomes dos vendedores responsáveis por cada venda (mesmo tamanho da lista de vendas); e (c) uma lista com os nomes de todos os vendedores da equipe.
Peça:

- Calcular o total de vendas da semana (somatório dos valores) e a venda média.
- Determinar qual foi a maior venda e quem a realizou (pode supor que as listas estão sincronizadas por índice, ou seja, posição das listas correspondem à mesma venda).
- Gerar um dicionário de total de vendas por vendedor (chave: nome do vendedor, valor: soma das vendas dele).
- A partir do dicionário, extrair o nome do vendedor com maior faturamento total na semana e o valor correspondente.
- Identificar quais vendedores (da lista de todos os vendedores) não realizaram nenhuma venda na semana (ou seja, não aparecem nas entradas de vendas) e mostrar seus nomes.

Aplicação: Esse exercício reúne operações numéricas (soma, média), uso de listas paralelas, condicional (para comparar vendas), dicionário (para agregação por pessoa), e até tuplas se quiser retornar pares. É bem contextualizado no cotidiano de um time de vendas. Terão que usar laços *for* para somar, para preencher o dicionário, e condicionais para máximos e para checar vendedores sem vendas (podem usar if vendedor not in dicionario ou valor zero).

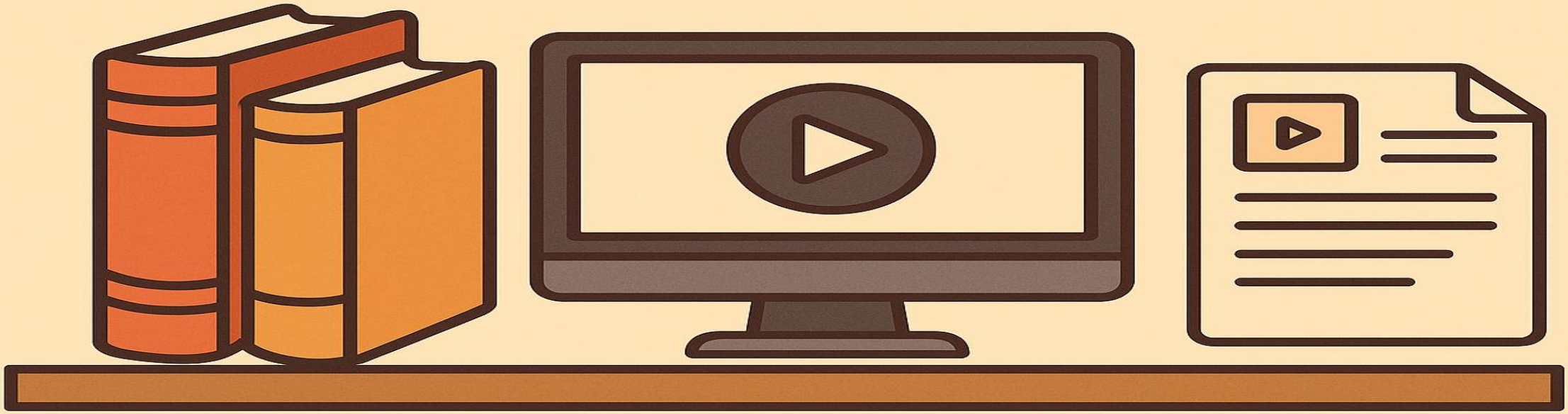
Atividade com IA



- Utilize a IA para corrigir os desafios acima
  - Conte com o apoio da IA para saber se o que você fez está correto.

- Faça individualmente...
  - *O seguinte desafio/exercício " <<digite aqui o exercício>> ", eu resolvi da seguinte forma*
    << cole aqui seu código>>
    Pode verificar se minha resposta está correta?
    Sugira modificações para melhorar a legibilidade e coerência do código. Explique as alterações, caso existam.

Próxima Aula



- Ler os conteúdos da semana 11 e ver os vídeos indicados.
- No curso online, ver a videoaula correspondente!

Boa semana e bons estudos!!