



P Y T H O N & S U A S A P L I C A Ç Õ E S

---

# ESTRUTURAS DE DADOS, MANIPULAÇÃO DE INFORMAÇÕES, INTRODUÇÃO A BIBLIOTECAS E AO GOOGLE COLAB.

POR JOÃO VÍTOR PAMPONET ESTEVES

# O QUE IREMOS APRENDER

---

1

Introdução ao Google Colab

2

Estruturas de dados

Listas

Tuplas

Dicionários

3

Biblioteca matplotlib

# INTRODUÇÃO AO GOOGLE COLAB

---



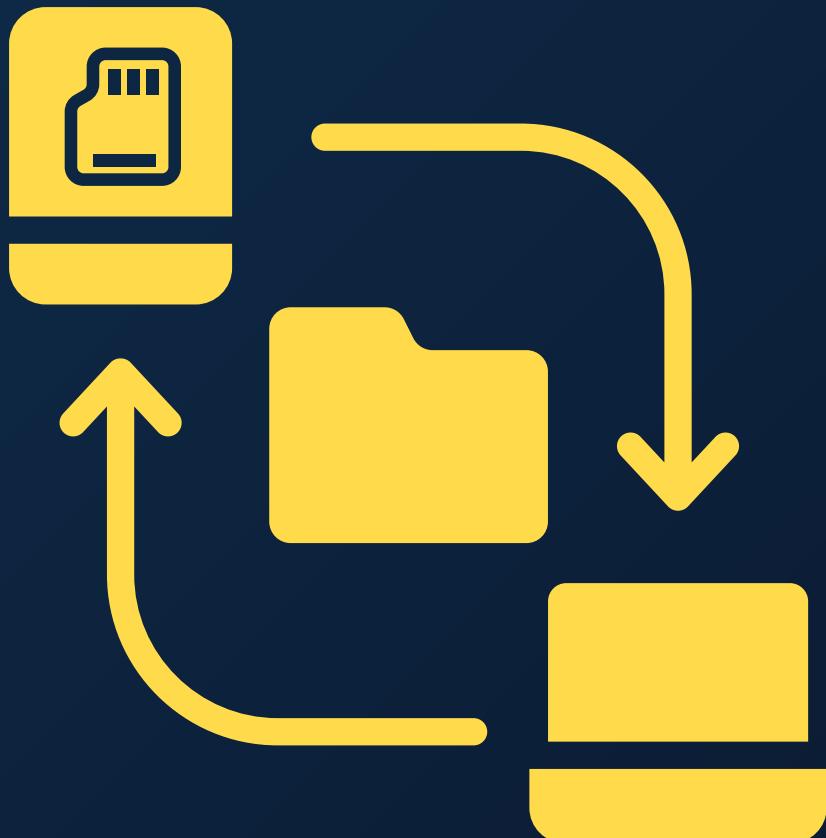
O Google Colab é uma plataforma online gratuita oferecida pelo Google que permite programar em Python diretamente do navegador, sem necessidade de instalação de softwares no computador.

Além de ser acessível, ele oferece integração com a nuvem, o que significa que os códigos e resultados ficam salvos online e podem ser acessados de qualquer dispositivo.

Outra grande vantagem é que o Colab já vem com diversas bibliotecas pré-instaladas, facilitando o uso em áreas como engenharia, ciência de dados e inteligência artificial.

# ESTRUTURAS DE DADOS

---



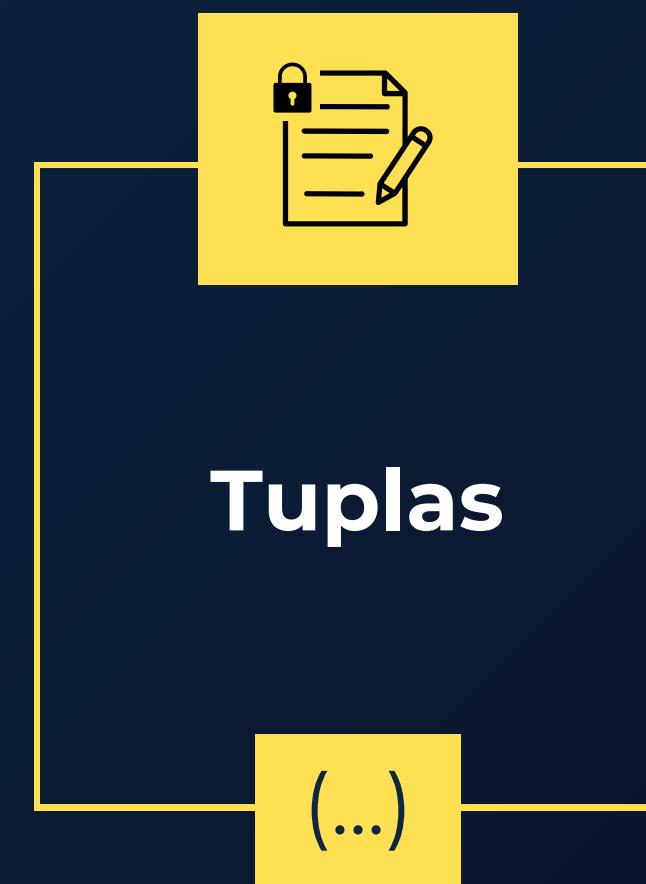
Estruturas de dados são formas organizadas de armazenar e gerenciar informações na memória do computador.

Elas definem como os dados são organizados e quais operações podem ser realizadas sobre eles.

# ESTRUTURAS DE DADOS

---

As estruturas de dados mais utilizadas em Python incluem:



Cada uma oferece diferentes vantagens de acordo com  
a necessidade de manipulação e acesso aos dados.

# ESTRUTURAS DADOS LISTAS

---

[...]

1.  \_\_\_\_\_
2.  \_\_\_\_\_
3.  \_\_\_\_\_
4.  \_\_\_\_\_
5.  \_\_\_\_\_

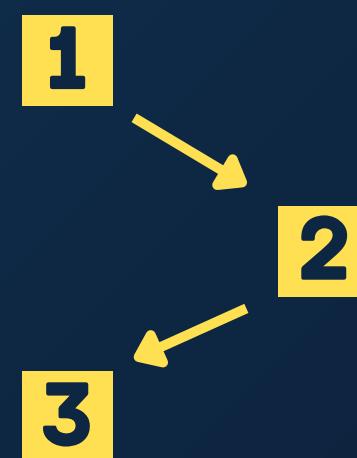
As listas em Python são usadas para armazenar múltiplos valores em uma única variável.

Elas permitem modificar seus elementos, adicionando, removendo ou alterando valores conforme a necessidade. São extremamente úteis para organizar informações de forma sequencial.

Um exemplo simples do dia a dia é a lista de compras do supermercado: podemos guardar os itens em uma lista, acessar cada um pelo índice e até adicionar novos produtos conforme lebramos.

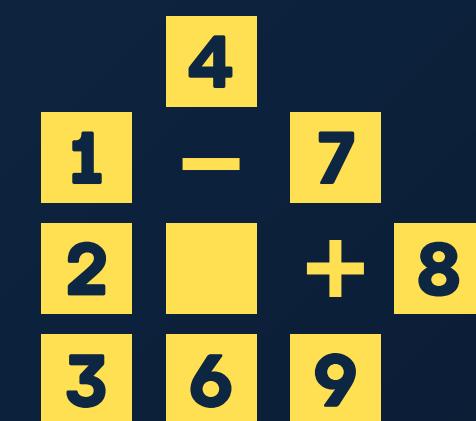
# CARACTERISTICAS DAS LISTAS

---



## Ordenada

Mantém a ordem de inserção



## Mutável

Pode alterar, adicionar, remover elementos



## Permite Duplicatas

Permite a inserção de itens duplicados



## Acesso por índice numérico

Minha\_lista[0]

# COMANDOS DE MANIPULAÇÃO DAS LISTAS

---

**Adiciona um item no final da lista**

**.append**

```
minha_lista = [10, 20, 30]
```

```
minha_lista.append(40)
```

```
print (minha_lista) = [10, 20, 30, 40]
```

**Remove um item da lista**

**.remov**

```
minha_lista = [10, 20, 30, 40, 50]
```

```
minha_lista.remove(30)
```

```
print (minha_lista) = [10, 20, 40, 50]
```

# PERCORRER LISTAS UTILIZANDO FOR

---

O **loop for** é a ferramenta ideal para percorrer automaticamente todos os elementos de uma lista, permitindo executar operações em cada item sem necessidade de controle manual de índices.

```
minha_lista = ["parafuso", "porca", "arruela"]  
  
for elemento in minha_lista:  
  
    print elemento
```

## saída

```
parafuso  
porca  
arruela
```

# ESTRUTURAS DADOS **TUPLAS**

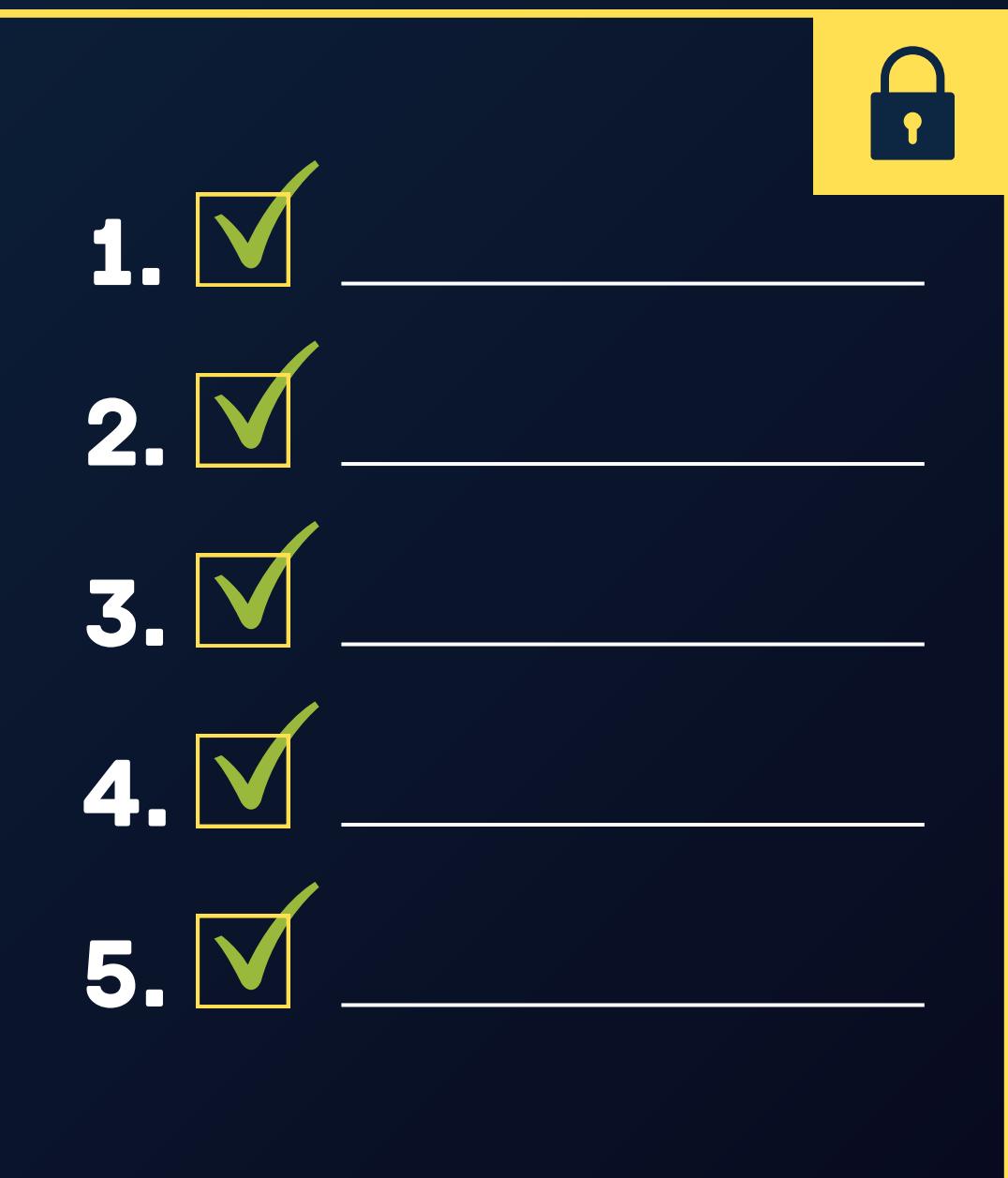
---

(...)

As tuplas são muito parecidas com as listas, mas possuem uma característica importante: são imutáveis, ou seja, não podem ser alteradas após sua criação.

Isso as torna ideais para representar dados que não devem ser modificados ao longo do programa.

Um exemplo cotidiano é o uso de tuplas para representar coordenadas geográficas (latitude e longitude) ou dimensões de um objeto (altura, largura e comprimento).



1.  \_\_\_\_\_

2.  \_\_\_\_\_

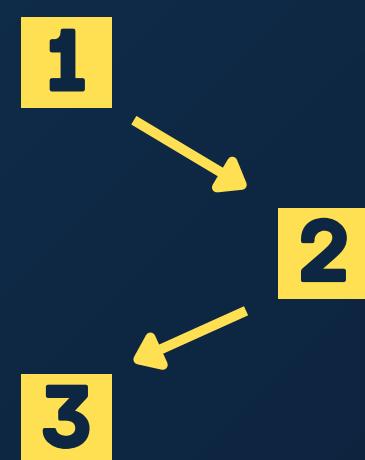
3.  \_\_\_\_\_

4.  \_\_\_\_\_

5.  \_\_\_\_\_

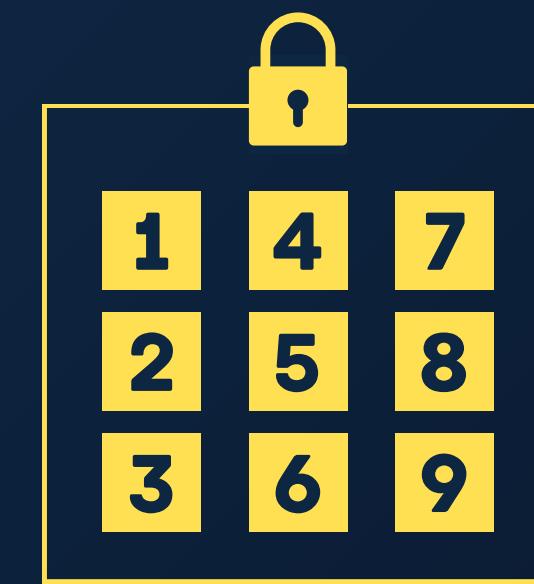
# CARACTERISTICAS DAS TUPLAS

---



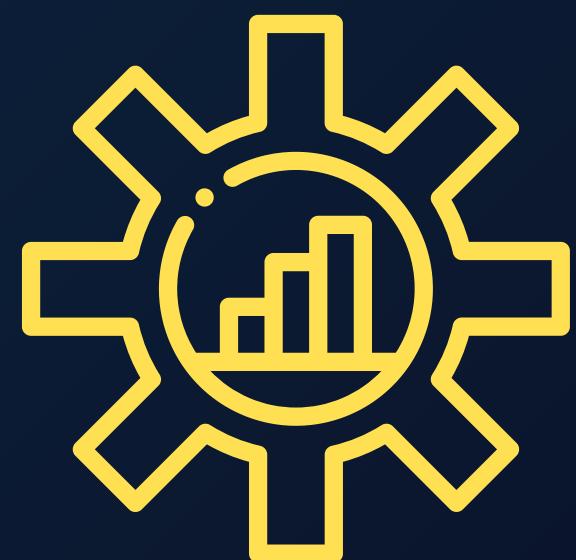
## Ordenada

Mantém a ordem



## Imutável

Não pode ser alterada  
após criação



## Mais eficiente

Em comparação com  
listas em termos de  
performance



## Segura

Para dados  
constantes

# ESTRUTURAS DADOS DICIONÁRIOS

---

{...}



Os dicionários em Python são estruturas de dados que armazenam informações em pares chave: valor, permitindo organizar dados de forma mais descritiva e intuitiva.

Eles funcionam como uma espécie de "tabela" ou "cadastro" em que a chave é usada para identificar e acessar rapidamente o valor associado.

Um exemplo prático do cotidiano é uma agenda telefônica: o nome da pessoa funciona como a chave, e o número de telefone como o valor.

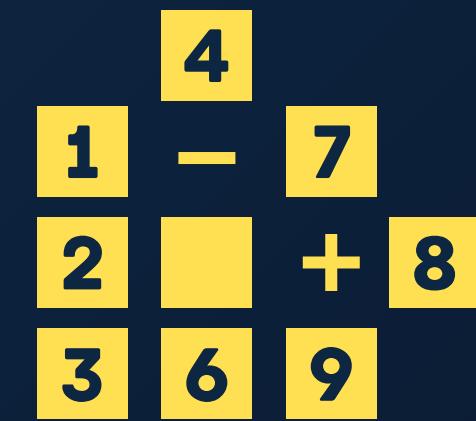
# CARACTERISTICAS DOS DICIONÁRIOS

---



## Ordenada

Mantém a ordem  
de inserção



## Mutável

Pode alterar, adicionar,  
remover elementos



## Acesso por chaves

Não por índices  
numéricos



## Chaves únicas

Não permite  
duplicatas

# CRIAR E ACESSAR DICIONÁRIO

---

## Criar Dicionário

```
material - {  
    "nome": "Aço Inox",  
    "densidade": 7.9,  
    "resistencia": 500  
}
```

## Acessar Valores

```
print(material["nome"])  
print(material.get("densidade"))
```

# COMANDOS DE MANIPULAÇÃO DO DICIONÁRIO

---

```
material = {"nome": "Aço Inox", "densidade": 7.9}
```

```
material["resistencia"] = 500
```

• Adiciona nova chave

```
material["densidade"] = 8.1
```

• Modifica valor existente

```
material.update({"preco": 25.0, "cor": "prata"})
```

• Adiciona múltiplos

# BIBLIOTECA MATPLOTLIB

---

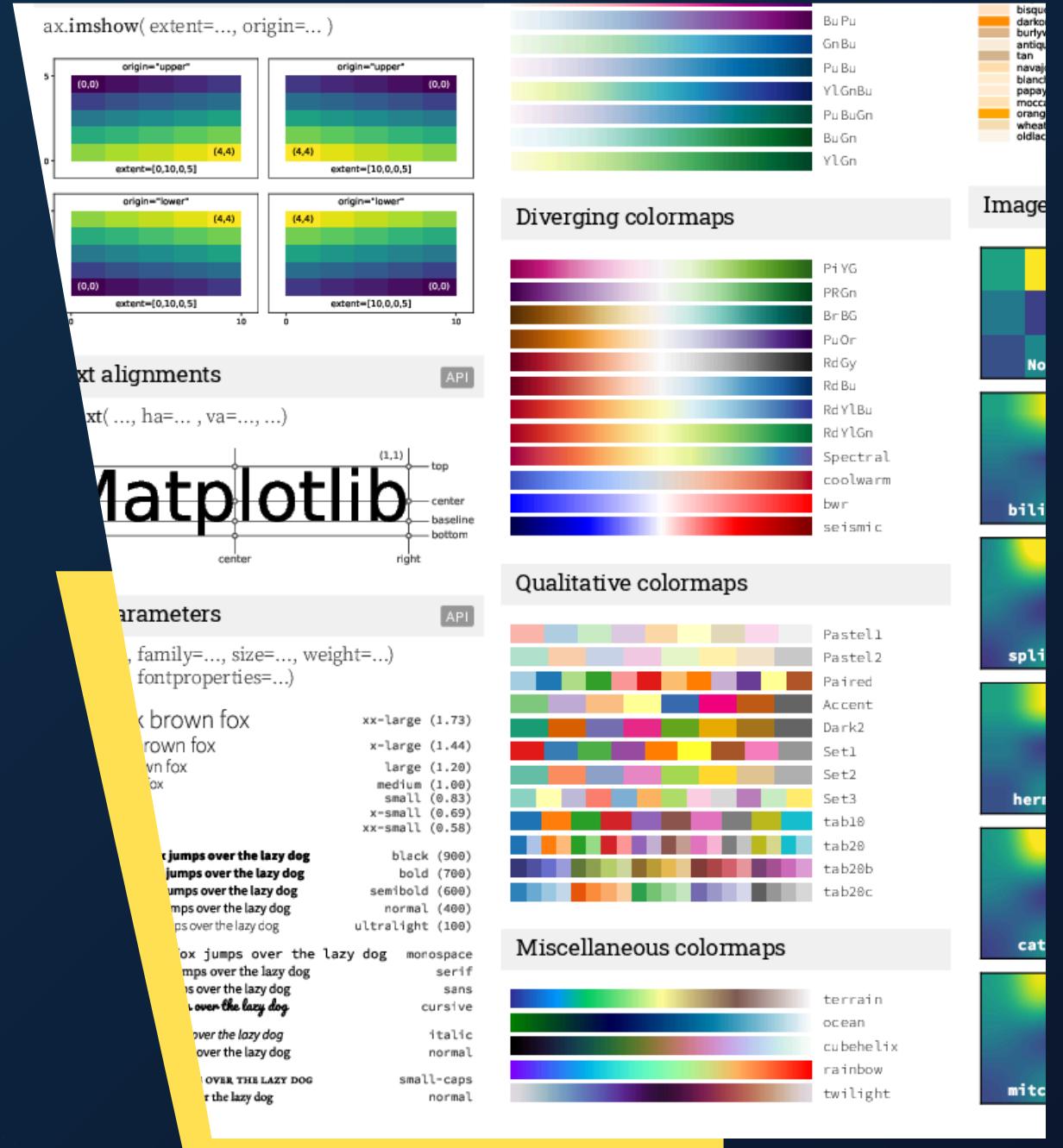


A matplotlib é uma das bibliotecas mais utilizadas em Python para criação de gráficos e visualização de dados.

Seu objetivo principal é **transformar informações numéricas em representações visuais**, o que facilita a análise e a interpretação de resultados.

Ela é especialmente útil em áreas como engenharia, ciência de dados e pesquisa, onde muitas vezes lidamos com grandes conjuntos de números que, sozinhos, são difíceis de compreender.

# UTILIDADE DA BIBLIOTECA MATPLOTLIB



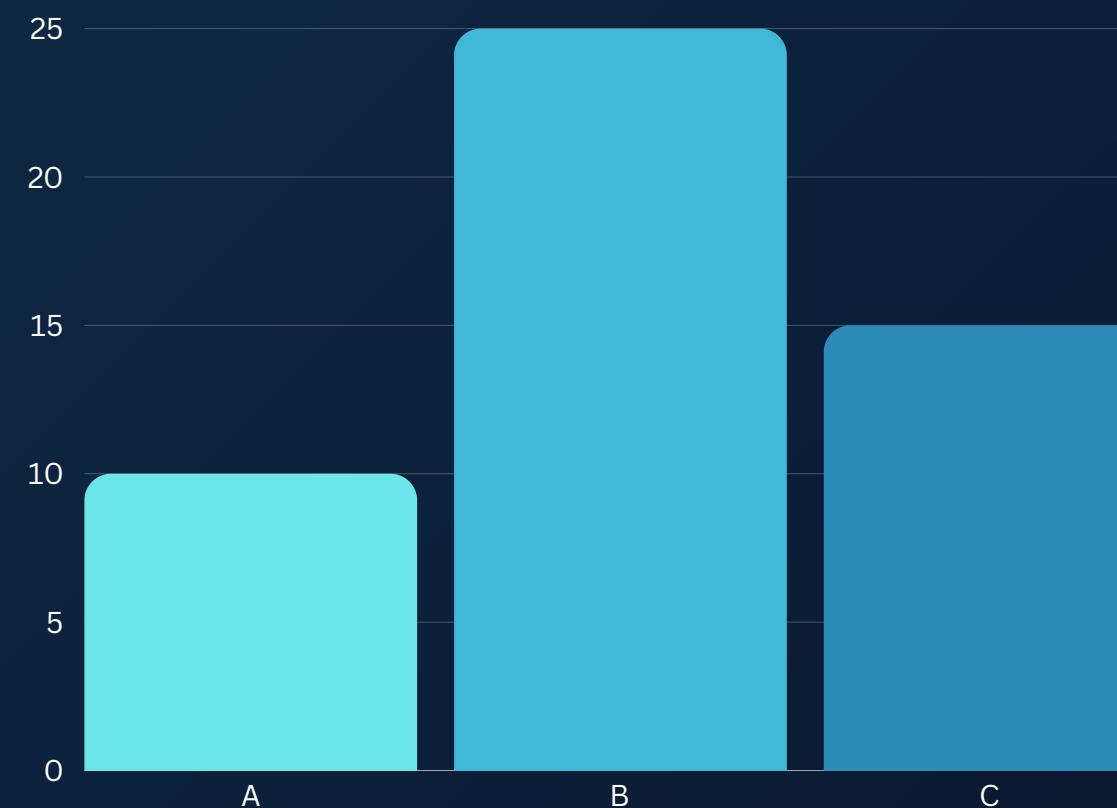
Uma das maiores vantagens do matplotlib é a sua versatilidade: **ele** permite criar gráficos de linhas, barras, dispersão, histogramas e até mesmo figuras tridimensionais.

Isso dá ao usuário a liberdade de escolher a melhor forma de representar cada tipo de informação.

Além disso, é possível personalizar todos os elementos dos gráficos, como cores, estilos de linha, títulos, legendas e rótulos, criando visualizações claras e adaptadas às necessidades do projeto.

# CRIANDO GRÁFICOS MATPLOTLIB

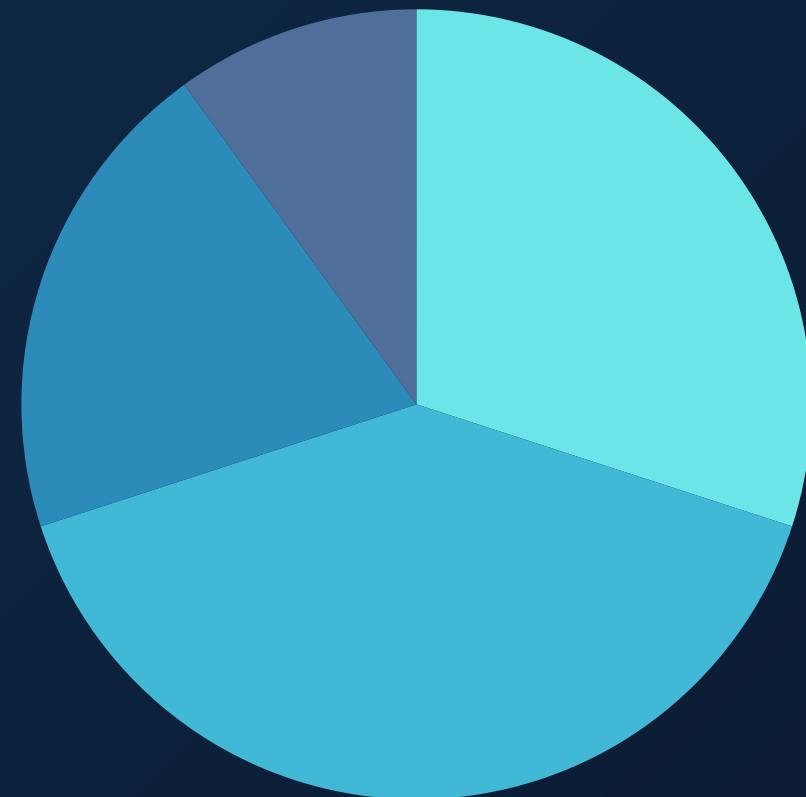
## GRÁFICO DE BARRAS SIMPLES



```
import matplotlib.pyplot as plt  
  
categorias = ['A', 'B', 'C']  
  
valores = [10, 25, 15]  
  
plt.bar(categorias, valores)  
plt.show()
```

# CRIANDO GRÁFICOS MATPLOTLIB

## GRÁFICO DE PIZZA SIMPLES



```
import matplotlib.pyplot as plt  
  
valores = [30, 40, 20, 10]  
rotulos = ['A', 'B', 'C', 'D']  
  
plt.pie(valores, labels=rotulos)  
  
plt.show()
```

# EXERCÍCIOS PRÁTICOS

---

## CONTROLE DE PRODUÇÃO

Crie um programa que gerencie uma lista de peças produzidas. O programa deve:

- 01** Permitir adicionar novas peças à lista
- 02** Mostrar todas as peças em produção
- 03** Remover uma peça específica quando ela for concluída
- 04** Exibir o total de peças atual

# EXERCÍCIOS PRÁTICOS

---

## DIMENSÕES DE PRODUTO

Crie um programa que trabalhe com as dimensões fixas de um produto (comprimento, largura, altura). O programa deve:

- 01** Armazenar as dimensões em tupla
- 02** Calcular o volume do produto
- 03** Verificar se o produto é considerado “Pequeno” ( $\text{volume} < 1000 \text{ cm}^3$ ) ou “Grande”
- 04** Exibir todas as dimensões de forma organizada

# EXERCÍCIOS PRÁTICOS

---

## CADASTRO DE MATERIAIS

Desenvolva um sistema de cadastro de materiais usando dicionário. O programa deve

- 01** Armazenar materiais com código único como chave
- 02** Cada material deve ter: nome, quantidade em estoque e preço unitário
- 03** Permitir consultar um material pelo código
- 04** Calcular o valor total em estoque de um material (quantidade x preço)

**ATÉ A PROXIMA**

---

