FICHA TRABALHO 1

MODALIDADE:	Aprendizagem +	Não aplicável	
CURSO:	Técnico/a Especialista em Gestão de Informação e Ciência dos Dados		
UFCD:	Visualização de dados em Python	CÓDIGO UFCD:	10809
FORMADOR/A:	Bruno Silva	DATA:	

OBJETIVOS

- Introdução a biblioteca Matplotlib
- Instalação e configuração da biblioteca
- Tipos de gráficos

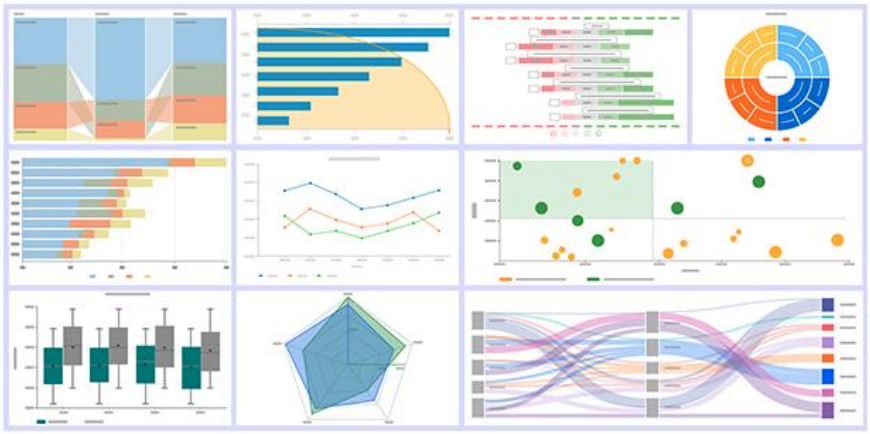
Biblioteca *Matplotlib*

Os gráficos ajudam a entender melhor o problema em estudo e as características dos dados, de modo a podermos extrair informações relevantes para a tomada de decisões.

A visualização de dados em Python tem sido cada vez mais essencial para análise e compreensão de informações em diversas áreas, desde negócios até ciência de dados. Ferramentas como o Matplotlib são extremamente populares, já que permitem a criação de gráficos informativos com poucos comandos.

Em ciência de dados são comuns os seguintes tipos de gráficos para representar dados dos mais diversos tipos, tais como:

- gráfico de linhas (line plot);
- gráfico de barras (bar chart);
- histograma;
- diagrama de caixa;
- gráfico de dispersão;
- gráfico de área;
- mapa de calor;
- gráfico de setores ou de pizza;
- gráfico de bolhas;
- entre outros ...



As bibliotecas de python usadas na construção de gráficos são o Matplotlib (uma das mais usadas depois oferece muitas opções de personalização), o Seaborn (alternativa ou Matplotlib para produzir gráficos mais

atraentes com menos linhas de código), o Pandas (permite preparar a informação, realizar a limpeza de dados e respetivos tratamentos para construir gráficos através de funções integradas em Dataframes).

Além disso, esta biblioteca é extremamente flexível, possibilitando a personalização completa dos gráficos, desde colocar identificações, exportar dados, entre outros.

A biblioteca também é muito compatível com outras ferramentas e pacotes de Python, como NumPy e Pandas, o que facilita o uso de grandes conjuntos de dados.

Instalação e configuração da biblioteca Matplotlib

Para instalar a biblioteca do Matplotlib temos de fazer os seguintes passos:

Página Mapplotlib: https://matplotlib.org/stable/install/index.html

Comando instalação Mapplotlib: https://matplotlib.org/

- **Windows:** pip install -U matplotlib
- **MacOS:** pip3 install -U matplotlib

Para trabalhar com a biblioteca Matplotlib, temos de importar a biblioteca no topo do vosso ficheiro. **No laboratório do jupyter,** comece por a pasta 10809 e de seguida **a pasta com o nome ficha_1**. Na nova pasta, vamos criar um novo ficheiro com o nome "**ficha_1.py**" e faça a importação da biblioteca na 1ªlinha de código:

```
#importar a biblioteca matplotlib
import matplotlib.pyplot
```

Mas, no código não queremos chamar por todo o nome da biblioteca. Como tal, **vamos criar um nome mais curto** que vai **representar tudo o nome da biblioteca**, ajudando na hora da invocação.

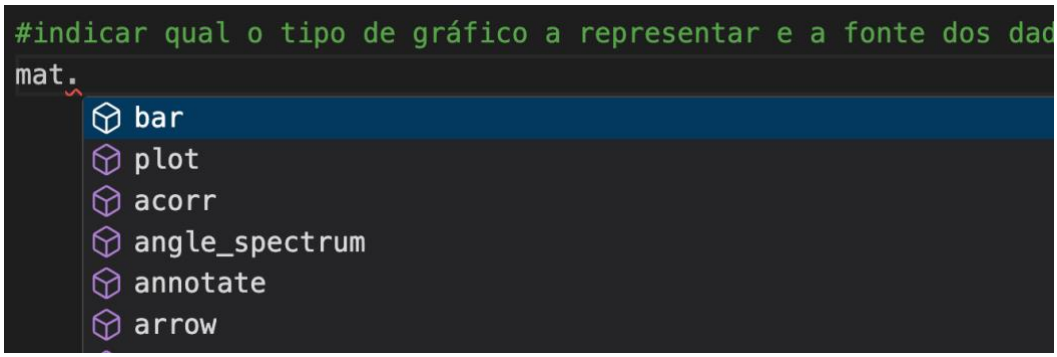
Como tal, ***a frente*** do nome da biblioteca podem colocar a palavra reservada **as** com o nome alternativo a frente:

import nome_biblioteca_a_usar **as** nome

```
#importar a biblioteca matplotlib
import matplotlib.pyplot as mat
```

Muito Importante: o nome mat foi apenas sugestivo para a demonstração.

Quando invocamos a biblioteca no código, reparem que vai aparecer um conjunto de gráficos que podemos arquitetar. **Para este exemplo, basta chamar o nome abreviado da biblioteca e indicar que queremos construir um gráfico de linha (mas tem outras opções):**

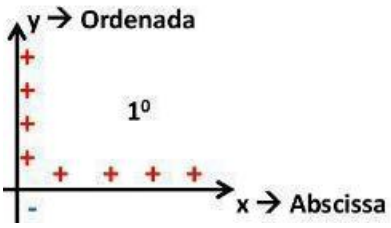


1. *Gráfico de linhas (line plot)*

O gráfico de linhas permite projetar a informação de dois vetores de dados (que vão indicar os pontos de interseção) e que vão ser destruídos pelas abcissas e as ordenadas.

Neste exemplo, vamos criar duas listas de informação:

```
#criação das listas das abcissas e ordenadas
lista_x = [1, 2, 3] #1ª lista (abcissas)
lista_y = [4, 5, 6] #2ª lista (ordenadas)
```



Após isso, vamos invocar a função do gráfico de linhas chamado **plot**() e este tem **2 parâmetros**:

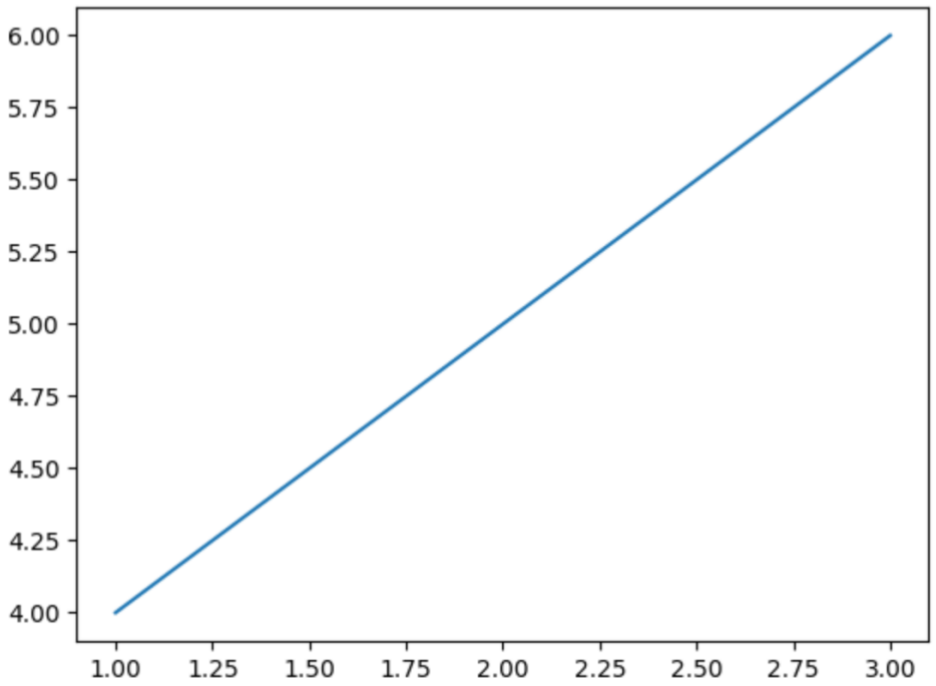
- **Parâmetro 1:** o valor das abcissas (neste caso a variável lista_x);
- **Parâmetro 2:** o valor das ordenadas (neste caso a variável lista_y);

```
#indicar qual o tipo de gráfico a representar
#e a fonte de dados a representar
mat.plot(lista_x, lista_y)
```

Depois de feitos os passos anteriores, deverá ter o seguinte resultado:

```
#indicar qual o tipo de gráfico a representar
#e a fonte de dados a representar
mat.plot(lista_x, lista_y)
```

```
[<matplotlib.lines.Line2D at 0x135b08f50>]
```

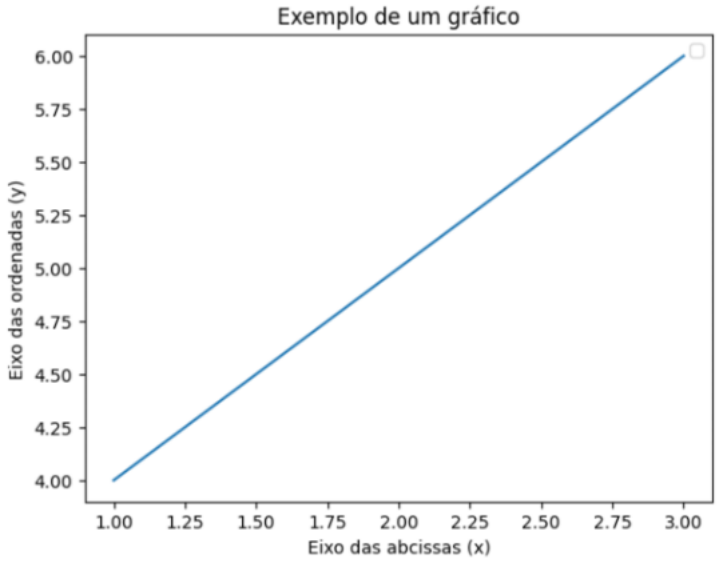


O gráfico apenas exibiu os valores. Como tal, falta personalizar o mesmo, como colocar os nomes dos eixos das abcissas, das ordenadas, nome do gráfico, entre outras informações.

- **Colocar o título do gráfico:** nome_biblioteca.**title**(“Exemplo gráfico”);
- **Colocar o título no eixo das abcissas:** nome_biblioteca.**xlabel**(“Eixo X”);
- **Colocar o título no eixo das ordenadas:** nome_biblioteca.**ylabel**(“Eixo Y”);

Como tal, devem colocar essa informação antes de exibir os dados com o comando do show():

```
#Personalizações
mat.title("Exemplo de um gráfico") #Colocar o texto no topo do gráfico
mat.xlabel("Eixo das abcissas (x)") #Colocar texto no eixo das abcissas
mat.ylabel("Eixo das ordenadas (y)") #Colocar texto no eixo das ordenadas

#indicar qual o tipo de gráfico a representar
#e a fonte de dados a representar
mat.plot(lista_x, lista_y, label="linha")
```

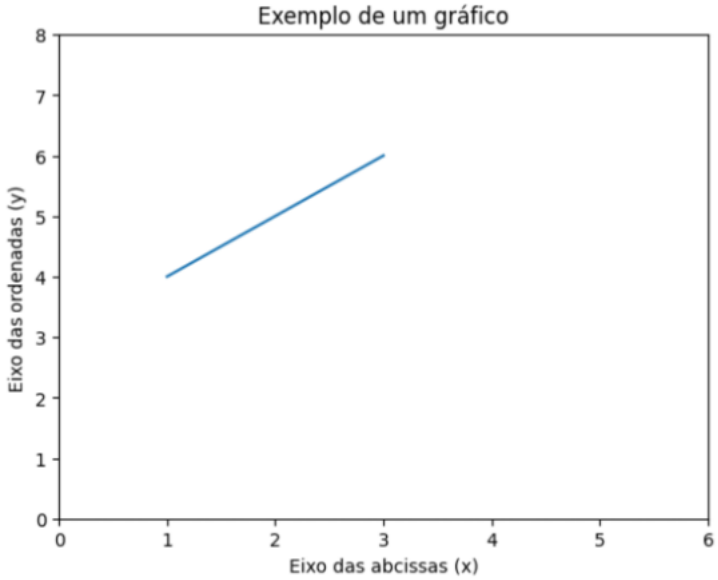


Mudar a escala do gráfico

Para mudar a escala dos valores dos eixos x e y, vamos utilizar as propriedades:

- **xlim → define** o limite **mínimo** e **máximo** do **eixo das abcissas (x)**
  - **Exemplo:** xlim(mínimo, máximo) → **xlim(0,8)**
- **ylim → define** o limite **mínimo** e **máximo** do **eixo das ordenadas (y)**
  - **Exemplo:** ylim(mínimo, máximo) → **ylim(0,8)**

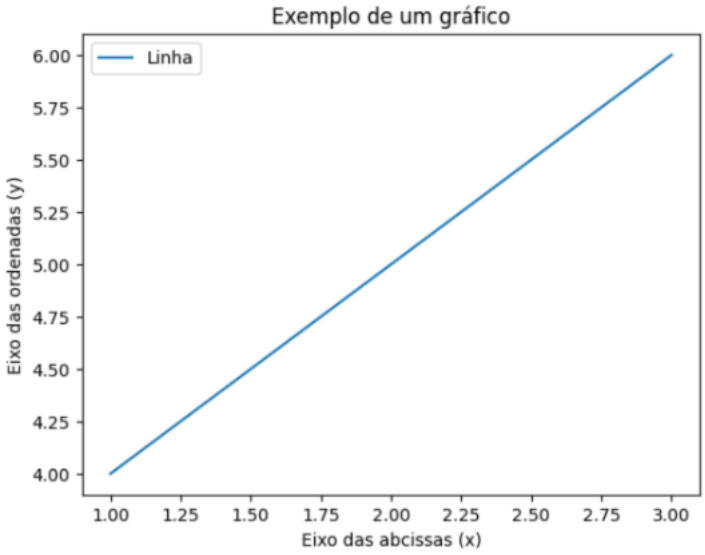
```
#Personalizações
mat.title("Exemplo de um gráfico") #Colocar o texto no topo do gráfico
mat.xlabel("Eixo das abcissas (x)") #Colocar texto no eixo das abcissas
mat.ylabel("Eixo das ordenadas (y)") #Colocar texto no eixo das ordenadas

mat.xlim(0,6) #define o limite minimo e máximo do eixo das abcissas
mat.ylim(0,8) #define o limite minimo e máximo do eixo das ordenadas

#indicar qual o tipo de gráfico a representar
#e a fonte de dados a representar
mat.plot(lista_x, lista_y, label="linha")
```



Colocar legendas no gráfico (propriedade legend)

Para incluir legendas no gráfico, podemos usar a propriedade legend que permite identificar a linha que estamos a trabalhar. Como tal, basta colocar essa linha de código após a instrução do plot() (antes disso o mesmo não exibe):

```
#indicar qual o tipo de gráfico a representar
#e a fonte de dados a representar
mat.plot(lista_x, lista_y)
mat.legend(["Linha"]) #Colocar uma pequena caixa com a legenda da linha
```



Também pode indicar a localização da caixa da legenda com o parâmetro loc. Este pode ser colocado em várias posições. Podem colocar o número ou a expressão textual:

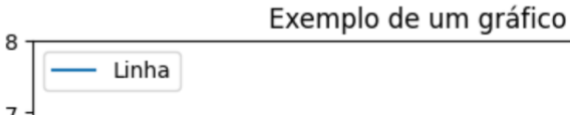
Location String	Location Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10

```
#indicar qual o tipo de gráfico a representar
#e a fonte de dados a representar
mat.plot(lista_x, lista_y)
mat.legend(["Linha"], loc = 2)
```

```
<matplotlib.legend.Legend at 0x137334190>
```


```
#indicar qual o tipo de gráfico a representar
#e a fonte de dados a representar
mat.plot(lista_x, lista_y)
mat.legend(["Linha"], loc = "upper left")
```

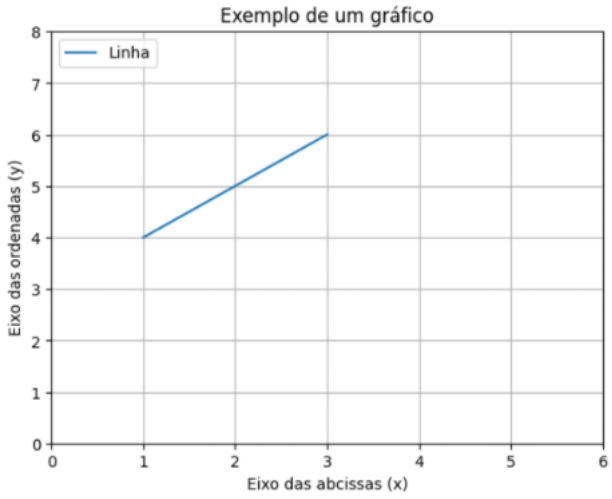
```
<matplotlib.legend.Legend at 0x13738ae90>
```



Colocar grelha no fundo dos gráficos (propriedade grid)

Para incluir a grelha quadriculada no fundo dos gráficos, basta apenas incluir a seguinte instrução:

```
mat.grid() #Coloca uma grelha por detrás da linha

#indicar qual o tipo de gráfico a representar
#e a fonte de dados a representar
mat.plot(lista_x, lista_y)
mat.legend(["Linha"], loc = "upper left")
```



Mostrar visualmente as coordenadas dos pontos

Para mostrar as informações dos pontos no gráfico, temos de usar a seguinte estrutura de repetição:

Código:

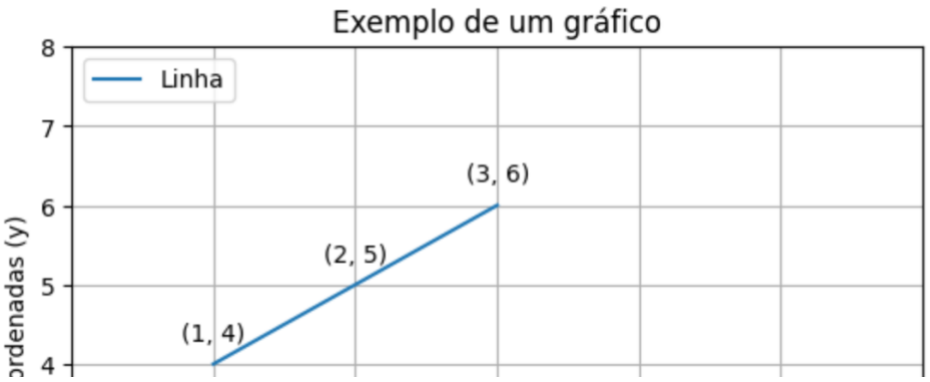
#colocar as informações dos pontos de interseção das duas listas

for i, (x, y) in enumerate(zip(lista_x, lista_y)):

mat.annotate(f"({x}, {y})", (x, y), textcoords="offset points", xytext=(0, 10), ha="center")

```
#colocar as informações dos pontos de interseção das duas listas
for i, (x, y) in enumerate(zip(lista_x, lista_y)):
    mat.annotate(f"({x}, {y})", (x, y), textcoords="offset points", xytext=(0, 10), ha="center")
#indicar qual o tipo de gráfico a representar
#e a fonte de dados a representar
mat.plot(lista_x, lista_y)
mat.legend(["Linha"], loc = "upper left")
```

```
<matplotlib.legend.Legend at 0x1373d3610>
```



Mudar o formato da linha do gráfico

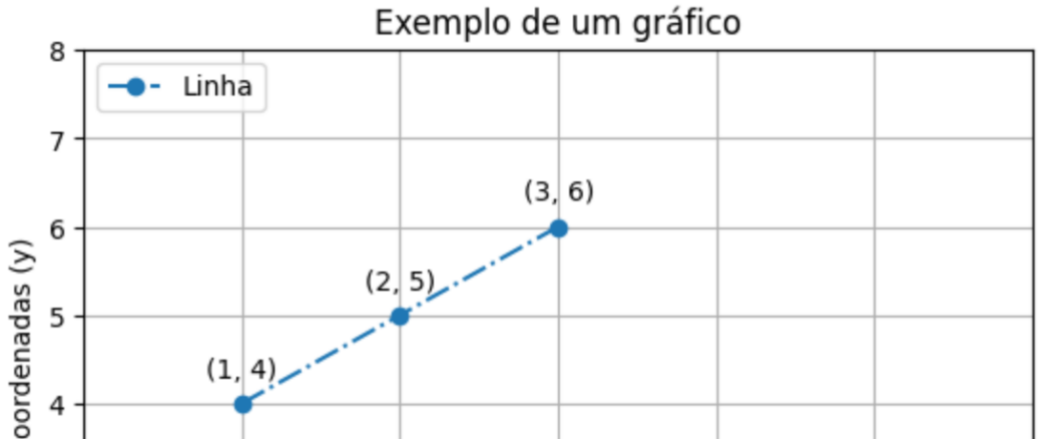
Para mudar o formato da linha, temos duas opções adicionais na propriedade plot. Como tal, pode usar as propriedades:

- **Marker** → coloca o símbolo no ponto de interseção (neste caso colocamos a palavra "o")
- **Linestyle** → indica qual o formato da linha a presentar (neste caso colocamos "-.")

```
#indicar qual o tipo de gráfico a representar
#e a fonte de dados a representar
mat.plot(lista_x, lista_y, marker="o", linestyle="-.")
```

```
#indicar qual o tipo de gráfico a representar
#e a fonte de dados a representar
mat.plot(lista_x, lista_y, marker="o", linestyle="-.")
mat.legend(["Linha"], loc = "upper left")
```

```
<matplotlib.legend.Legend at 0x13743c050>
```



Pode usar outros formatos no campo Linestyle (suportados pelo gráfico)

Com tal deve consultar o seguinte website que tem uma tabela resumida com os marcadores e símbolos:

https://www.w3schools.com/python/matplotlib_markers.asp

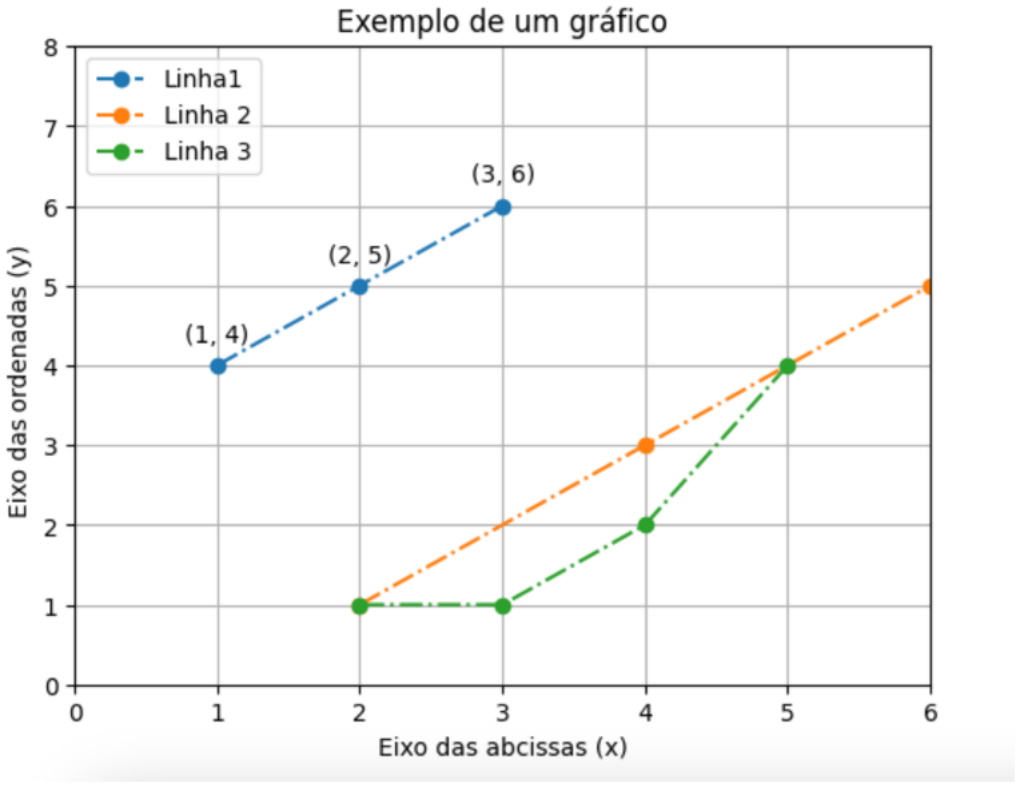
Criar gráfico de várias linhas

Para criar várias linhas independentes podemos declarar várias plots diferentes:

```
#criação das listas das abcissas e ordenadas
lista_x = [1, 2, 3] #Linha 1 (abcissas)
lista_y = [4, 5, 6] #Linha 1 (ordenadas)

lista_x2 = [2, 4, 6] #Linha 2 (abcissas)
lista_y2 = [1, 3, 5] #Linha 2 (ordenadas)

lista_x3 = [5, 4, 3, 2] #Linha 3 (abcissas)
lista_y3 = [4, 2, 1, 1] #Linha 3 (ordenadas)
```

```
#indicar qual o tipo de gráfico a representar
#e a fonte de dados a representar
mat.plot(lista_x, lista_y, marker="o", linestyle="-.")
mat.plot(lista_x2, lista_y2, marker="o", linestyle="-.")
mat.plot(lista_x3, lista_y3, marker="o", linestyle="-.")

mat.legend(["Linha1", "Linha 2", "Linha 3"], loc = "upper left")
```

```
<matplotlib.legend.Legend at 0x13795b390>
```



2. *Gráfico de barras (bar())*

Um **gráfico de barras** (bar) serve para comparar dados categóricos visualizando-os como barras retangulares, onde o comprimento ou altura de cada barra é proporcional ao valor que representa. É uma ferramenta eficaz para exibir padrões, identificar tendências, comparar diferentes categorias e facilitar a interpretação de informações.

Neste exemplo, **vamos criar as listas**:

- Nomes das frutas;
- Quantidade de cada fruta;
- Cores que queremos dar a cada fruta quando visualizar graficamente;

```
#dados para o gráfico
frutas = ["maças", "mirtilos", "peras", "laranjas"]
unidades = [40, 100, 35, 55]
cores_barras = ["tab:red", "tab:blue", "tab:green", "tab:orange"]
```

Em **termos de personalização**, vamos colocar as seguintes informações:

- **Título do gráfico** com o nome “Exemplo de um gráfico”;
- **Título do eixo das abcissas** “Frutas”;
- **Título do eixo das ordenadas** “Unidades”;

```
#Personalizações
mat.title("Exemplo de um gráfico") #Colocar o texto no topo do gráfico
mat.xlabel("Frutas") #Colocar texto no eixo das abcissas
mat.ylabel("Unidades") #Colocar texto no eixo das ordenadas
```

No final, vamos pedir **para construir o gráfico de barras (bar)** e colocar **4 parâmetros**:

- **Parâmetro 1** → qual os valores a colocar no eixo das abcissas (neste caso o nome das frutas);
- **Parâmetro 2** → qual os valores a colocar no eixo das ordenadas (neste caso os valores numéricos de cada fruta para construir cada barra individualmente);
- **Parâmetro 3** → colocar o parâmetro **label** e indicar que deve mostrar a informação das frutas;
- **Parâmetro 4** → colocar as cores personalizadas (neste caso utilizar a variável cores_barras)

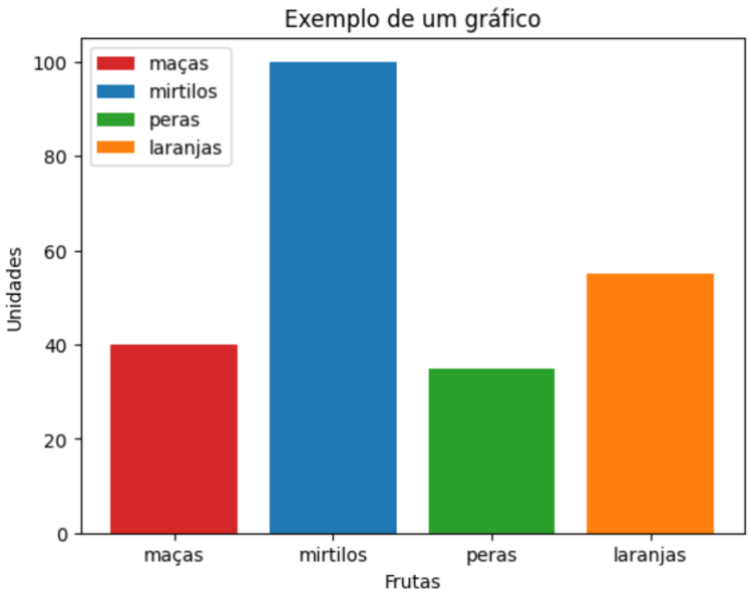
```
#indicar qual o tipo de gráfico a representar
#a fonte de dados a representar
mat.bar(frutas, unidades, label=frutas, color=cores_barras)
```

Para incluir legendas no gráfico, podemos usar a propriedade **legend** que permite identificar a linha que estamos a trabalhar. Como tal, basta colocar essa linha de código após a instrução do **bar()** (antes disso o mesmo não exibe) e passar 2 parâmetros:

- **Parâmetro 1** → qual os valores a colocar no eixo das abcissas (neste caso o nome das frutas);
- **Parâmetro 2** → qual os valores a colocar no eixo das ordenadas (neste caso os valores numéricos de cada fruta para construir cada barra individualmente);

```
#mostrar a legenda no gráfico
mat.legend(frutas, loc = "upper left")
```

Resultado:



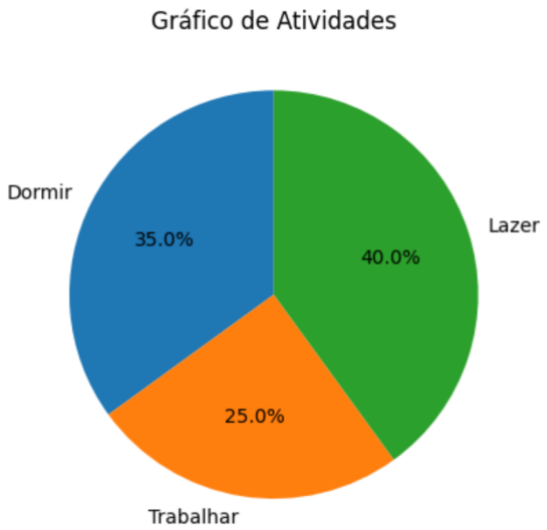
3. *Gráfico de radial (pie())*

Um gráfico radial serve para comparar várias categorias (ou variáveis) de forma visual e mostrar o desempenho de cada uma em relação a um ponto central. Ele é útil para analisar múltiplas variáveis simultaneamente, permitindo identificar rapidamente pontos fortes e fracos, tendências e padrões, como é o caso de um gráfico de teia de aranha ou radar.

Neste exemplo, estamos a criar um gráfico para dividir a percentagem das atividades de um dia, do qual a gráfico pie utiliza 4 parâmetros:

- **Parâmetro 1** → qual os valores
- **Parâmetro 2** → qual os textos para encaixar no parâmetro anterior;
- **Parâmetro 3** → qual o formato dos valores a exibir (neste caso estamos a indicar para colocar valores com 1 casa decimal e com o símbolo da percentagem);
- **Parâmetro 4** → em que ângulo começa a exibir os valores

```
#dados para o gráfico
fatias = [35, 25, 40]
atividades = ["Dormir", "Trabalhar", "Lazer"]

#construir gráfico circular pie()
mat.title("Gráfico de Atividades") #Colocar o texto no topo do gráfico
mat.pie(fatias, labels=atividades, autopct="%1.1f%%", startangle=90)
```



4. Gráfico de Histograma

Mostrar a distribuição de um conjunto de dados com a função hist()

Um histograma é um gráfico que exibe distribuições de frequência e o número de observações dentro de cada intervalo.

Exemplo: Temos um conjunto de dados sobre as alturas de peça a altura de 50 pessoas, e pretendemos ver os dados num histograma:

Passo 1 – Criar a variável com o nome **alturas** e vamos **armazenar uma lista de 50 alturas**:

Lista com 50 alturas entre 1.50 e 2.20 metros

alturas = [1.68, 1.75, 1.59, 1.92, 2.03, 1.84, 1.77, 2.11, 1.63, 1.56, 1.89, 2.00, 1.73, 1.61, 2.07, 1.66, 1.79, 1.94, 1.70, 1.58, 2.12, 1.87, 1.53, 1.97, 2.01, 1.76, 1.80, 1.65, 1.60, 2.14, 1.82, 2.18, 1.55, 1.69, 1.85, 1.62, 2.08, 1.90, 1.78, 2.04, 1.67, 2.16, 1.72, 1.74, 2.20, 1.57, 1.91, 1.64, 2.06, 2.10]

```
# Lista com 50 alturas entre 1.50 e 2.20 metros
alturas = [
    1.68, 1.75, 1.59, 1.92, 2.03, 1.84, 1.77, 2.11, 1.63, 1.56,
    1.89, 2.00, 1.73, 1.61, 2.07, 1.66, 1.79, 1.94, 1.70, 1.58,
    2.12, 1.87, 1.53, 1.97, 2.01, 1.76, 1.80, 1.65, 1.60, 2.14,
    1.82, 2.18, 1.55, 1.69, 1.85, 1.62, 2.08, 1.90, 1.78, 2.04,
    1.67, 2.16, 1.72, 1.74, 2.20, 1.57, 1.91, 1.64, 2.06, 2.10
]
```

Passo 2 – Criar as personalizações do gráfico:

```
#Qual o titulo do gráfico
mat.title('Histograma de Alturas')
#Qual o titulo do eixo do X
mat.xlabel('Altura (m)')
#Qual o titulo do eixo do Y
mat.ylabel('Frequência')
```

Passo 3 – Criar o gráfico do tipo histograma (hist()) e colocar os **4 parâmetros:**

- **Parâmetro 1 -** variável dos dados (neste caso das alturas)
- **Parâmetro 2 –** usar a propriedade bins e indicar o número de barras que deseja aparecer na análise do histograma (bins=10)
- **Parâmetro 3 -** cor das **bordas** de cada barra
- **Parâmetro 4 -** cor de **preenchimento** da barra

```
#construir gráfico histograma
mat.hist(alturas, bins=10, edgecolor='black', color='blue')
```