Matplotlib

- External library for creating visuals
- Has axes and figure

```
x = np.linspace(0, 2 * np.pi, 50) # Change from 5 to 500
fig, ax = plt.subplots()
ax.plot(x, np.sin(x)) # Line Chart
ax.plot(x, np.sin(x))
```

Q: How to distinguish which one is sine and which is cos [Try adding label in plot & legend. [label = in plot and plt.legend() later]
Q: What is value of sin(pi) or sin(3.14). Try adding grid (plt.grid()
Q: I want plot of degree value not radian
Q: I want xticks to be in space of 30. xticks = np.arrange(-360, 361, 30); plt.xticks(xticks)

Customizing

- Line plot is plotted by .plot
- We can customize the line plot with below arguments:
  - linewidth = number ➔ Sets thickness of line
  - label = text ➔ Adds label to plot. Later displayed on legend
  - color = ‘b’ ➔ Color of plot. Initial / valid name
  - marker = ‘o’ ➔ Markers. v, ^, <, >, + , d, x, *, .
  - linestyle = ‘—’ ➔ Dash line. -, --, -., :
  - alpha = 0.5 ➔ Set the transparency of plot
- We can combine color, marker and style as ‘ro—’

Customizing

- Chart without proper label in meaning less.
- We need to add label on axis, header for giving context
- Customization:
  - ax.set_xlabel('text', color, fontsize) ➔ Add label on x-axis
  - ax.set_ylabel('text') ➔ Add label on y-axis
  - ax.set_title('title', loc="left") ➔ Add title on plot
  - ax.set_xticks(list) ➔ Control Position of Ticks
  - ax.set_xticklabels(list, rotation, fontsize) ➔ Label, rotation and size of tick
  - ax.set_xscale("log") ➔ Set scale of x – axis as log
  - ax.set_xlim(40, 90) ➔ X axis is from 40 to 90
  - ax.spines["top"].set_visible(False) ➔ Hide to bar
    - Other values: top, bottom, right, left
    - Other methods: .set_color('red'), .set_linewidth(20)
  - fig.suptitle(text) ➔ Title for whole plot

Subplot

- For displaying multiple plot together at separate position

```
fix, ax = plt.subplots(2)  # or (1,2)
ax[0].plot(x, np.sin(np.radians(x)), label="sin")
ax[1].plot(x, np.cos(np.radians(x)), label="cos")
plt.legend() # loc='center' / 'upper right' / 'lower left'
plt.show()
```

Q: Where is legend? Use ax[0].legend() instead to show

Q: plt.tight_layout() to avoid axis overlapping

Barplot

- .bar or .barh is used for bar plot
- One has to be categorical & another numeric variable

Example:

```
df = pd.DataFrame({"country": ["NEP", "IND", "BAN"],
                   "Gold": [10, 25, 5], "Silver": [20, 10, 10]})

fig, ax = plt.subplots()

ax.bar(df["country"], df["Gold"]) # or barh

# We can add yerr= for adding error bar in plot
```

Stacked

- .bar or .barh is used for bar plot
- One has to be categorical & another numeric variable

Example:

```
df = pd.DataFrame({"country": ["NEP", "IND", "BAN"],
                   "Gold": [10, 25, 5], "Silver": [20, 10, 10]})

fig, ax = plt.subplots()
ax.bar(df["country"], df["Gold"]) # or barh
ax.bar(df["country"], df["Silver"], bottom=df["Gold"])

# What is stacked is required on horizontal plot? (use left= instead bottom)
```

Add Label

```
# barh
for i in range(len(df)):
    country = df["country"][i]
    gold = df["Gold"][i]
    silver = df["Silver"][i]
    ax.text(gold/2, country, str(gold), va='center', ha='center’, color='b')
    ax.text(
     gold+silver/2, country, str(silver), va="center", ha="center", color= "b")
```

How to add for total
How to add on vertical

Grouped

```
x = np.arange(len(df["country"]))  # positions for each country
bar_width = 0.35  # width of each bar

# Plot grouped bars
plt.bar(x - bar_width/2, df["Gold"], width=bar_width,
label="Gold", color="gold")
plt.bar(x + bar_width/2, df["Silver"], width=bar_width,
label="Silver", color="silver")

# Add labels and title
plt.xticks(x, df["country"])
```

Question: Add label to these stacked chart.

Pie/Donut Chart

```
ax.pie(x=df["Gold"], labels=df["country"], autopct="%1.1f%%")
# wedgeprops=dict(width=0.4) # linewidth, edgecolor


def autopct_format(values):
    def my_format(pct):
        total = sum(values)
        val = int(round(pct * total / 100.0))
        return f"{val} ({pct:.1f}%)"
    return my_format

# Pie chart
ax.pie(x=df["Gold"], labels=df["country"], autopct =
autopct_format(df["Gold"]), startangle=90,)
```

Histogram

- Show distribution of a value with frequency

Example:

```
ax.hist(data, label='height')
ax.hist(data_2, label='weight')
ax.legend()
plt.show()
```

Parameters:

histtype ➔ step if we don't want to fill the color inside
bins ➔ integer or list denoting number of bins or bin group
density ➔ Boolean indicating whether to normalize a histogram
Cumulative ➔ Boolean indicating cumulative plot or not

Box Plot

- .boxplot is used for Box plot

Example:

```
ax.boxplot(data_series)
ax.boxplot([series_1, series_2], labels=[“l1”, “l2”])
ax.legend()
```

Arugments:

patch_artist ➔ Enables fill
notch ➔ Boolean to indicate notch around median

boxprops=dict(facecolor=’red’, color=’red’)
whiskerprops=dict(color=’green’, linewidth=3, linestyle=‘—’)
capprops=dict(color’=’blue’),
medianprops=dict(color=’yellow’)
Orientation = “horizonal” # make plot horizontal

Scatter Plot

- .scatter
- For visualizing the correlation between variables

Example:

```
ax.scatter(x, y)
# Give color map with c
# Control size with s parameter
# edgecolor for control color of edge of bubble
# marker for controlling shape
# facecolors = “none”  if no fill
```

Style

- fig, ax = plt.subplots(1,2, figsize=(4,5)) # 4 inch wide

Need to specify GridSpec for different size subplot

- plt.style.use('ggplot')
  - Choosing theme in plot
  - Other parameters: dark_background, tableau-colorblind10, grayscale

Saving

- fig.savefig('file_name.png')
- fig.set_size_inches([3, 5])
- fig.savefig('pic.svg', dpi=500)

Call before plt.show()

Plot Image

```
from PIL import Image
import numpy as np

im = Image.open('name.png').convert('L')   # 'L' mode = grayscale
arr = np.array(im)  # arr.flipud
inv_arr = 255 – arr
bin_image = (arr > 127).astype(np.uint8) * 255

print(arr.shape)   # (height, width)
print(arr.dtype)   # usually uint8, values 0–255
plt.axis("off")

plt.imshow(arr, cmap='gray')
```