Day 11: Matplotlib basics, plots and customization techniques

📌 Topics Covered

- Customization in Matplotlib:
  - Markers
  - Adding labels
  - Configuring grid
  - Creating subplots
  - Styling plots
  - Resizing plots
  - Adjusting plot transparency
  - Changing fonts
  - Setting tick label font size
  - Changing plot background color
  - Moving axis labels

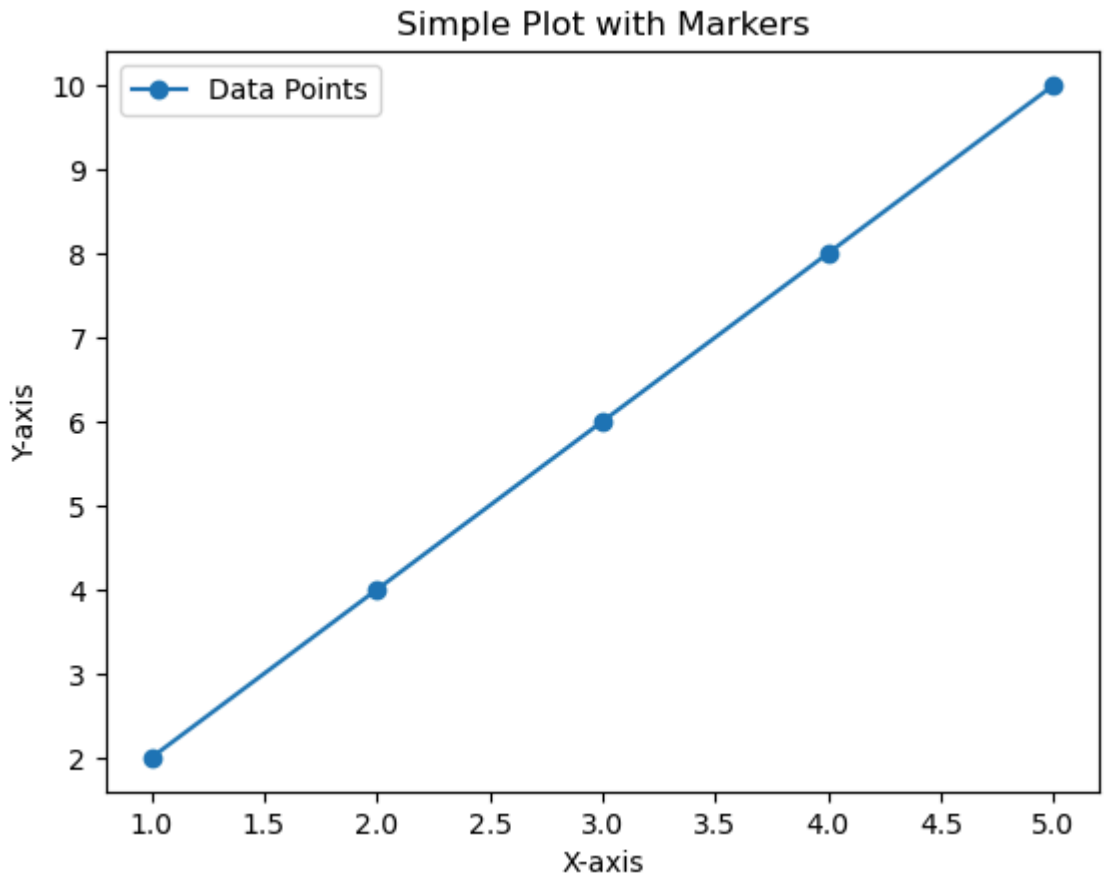
In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Customizations in Matplotlib

Matplotlib Markers

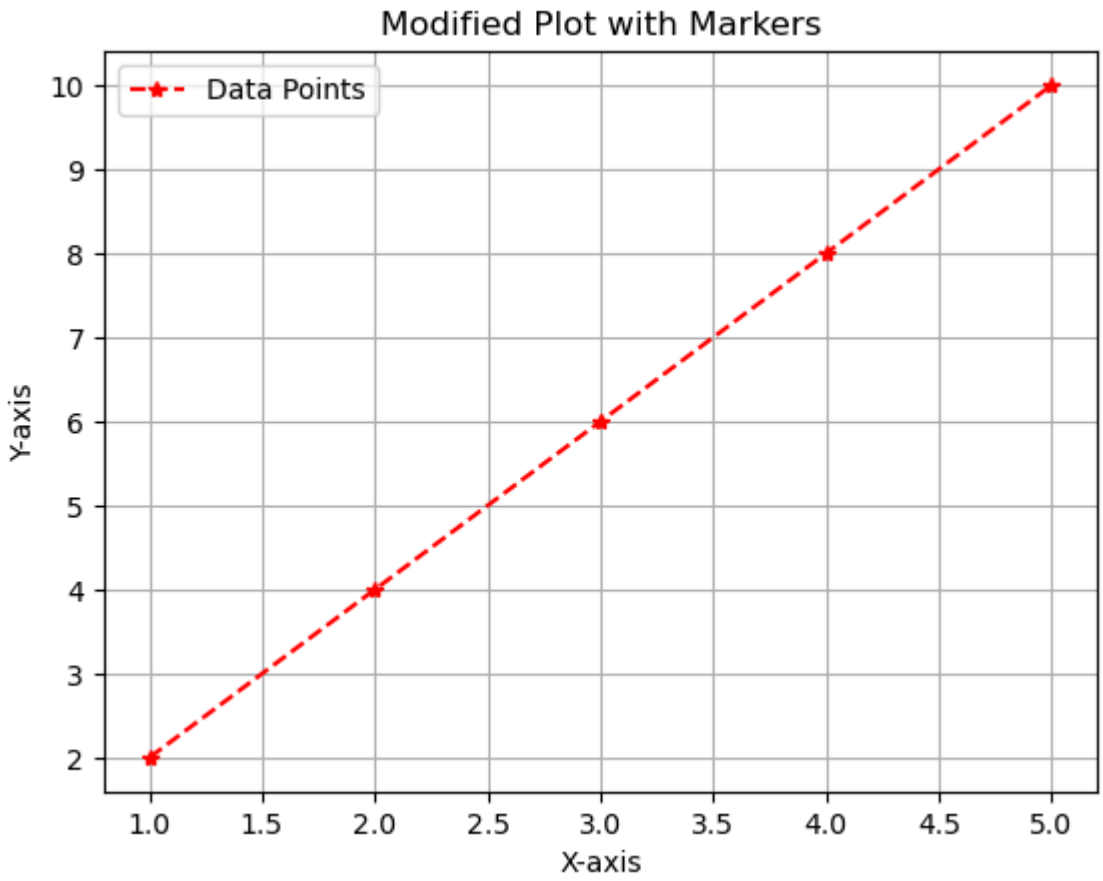
The markers module in Matplotlib helps highlight individual data points on plots, improving readability and aesthetics.

In [5]:
```
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
plt.plot(x, y, marker='o', linestyle='-', label='Data Points')
plt.title('Simple Plot with Markers')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.show()
```



In [2]:

```
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

plt.plot(x, y, marker='*', linestyle='--', color='r', label='Data
Points')
plt.title('Modified Plot with Markers')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.grid(True)
plt.show()
```



Different Marker styles

'.' Point (pixel) ',' Pixel (very small point) 'o' Circle 'v' Triangle down '^' Triangle up '<' Triangle left '>' Triangle right 's' Square 'p' Pentagon '*' Star 'h' Hexagon (pointing up) 'H' Hexagon (flat top) '+' Plus 'x' X 'D' Diamond 'd' Thin diamond

Line Style Refernce

Solid line '-'m Dashed line '‑‑' Dash-dot line '-.' Dotted line ':' Long dashed line '--' Custom dash 'dotted'

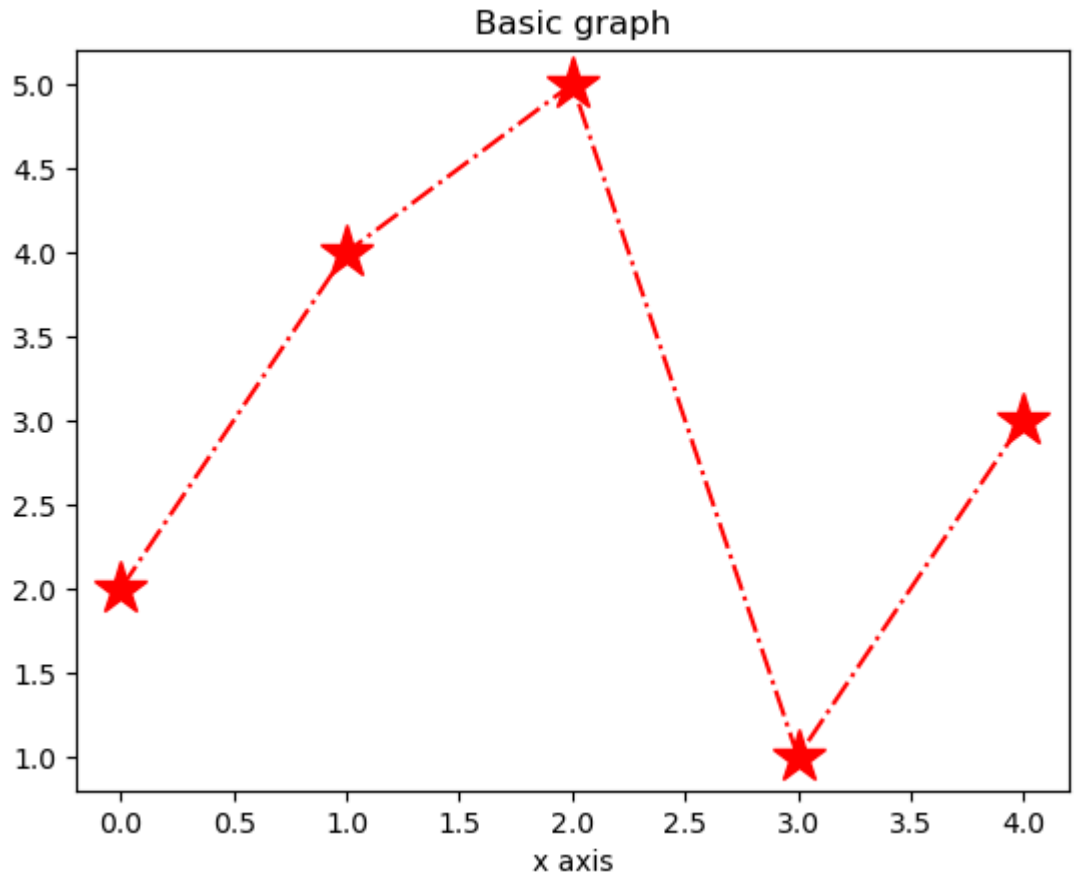
Colur style Refernce

Red 'r' Green 'g' Blue 'b' Cyan 'c' Magenta 'm' Black 'k' Yellow 'y' Orange 'orange' Purple 'purple' White 'w'

In [ ]:
```
# fmt Parameter in Matplotlib
the fmt parameter in Matplotlib is actually a combination of marker, line style, and col
```

In [5]:
```
x = [2, 4, 5, 1, 3]
plt.plot(x, '*-.r', ms = 20)  #ms = marker size
```

```
plt.title('Basic graph')
plt.xlabel("x axis")
plt.show()
```



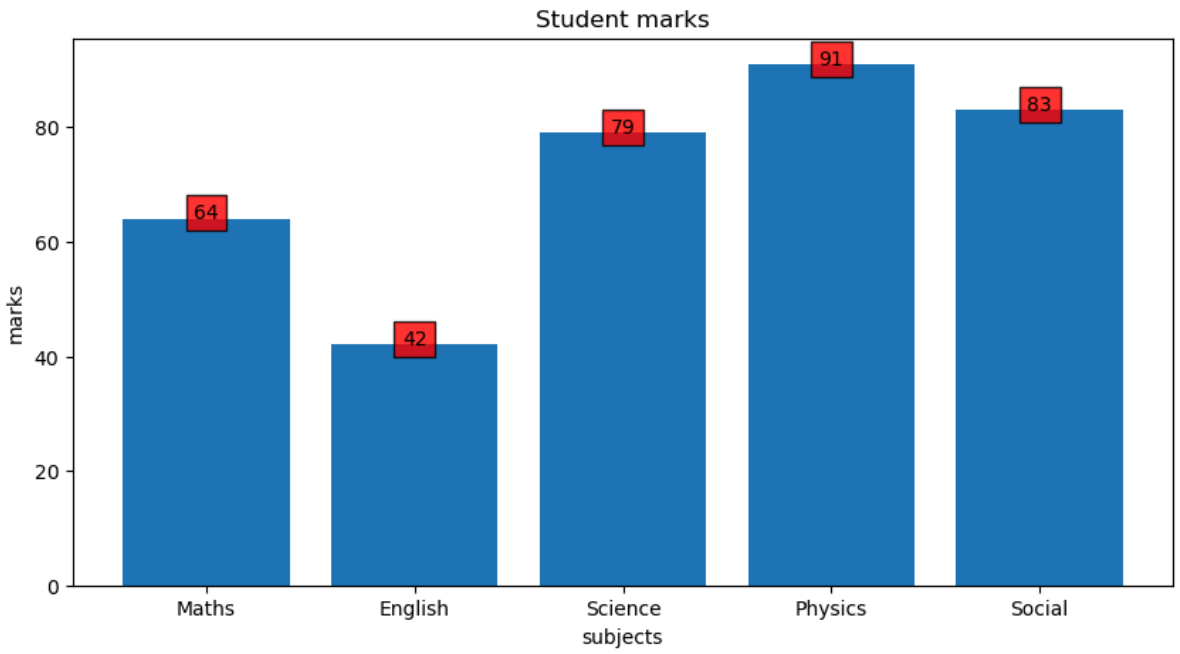
Adding value labels on a Matplotlib Bar Chart

Parameters: x, y: Coordinates where the text will be displayed. s: The string (value label) to be displayed. ha: Horizontal alignment ('center', 'left', or 'right'). bbox: A rectangular box surrounding the text for better visibility.

In [10]:

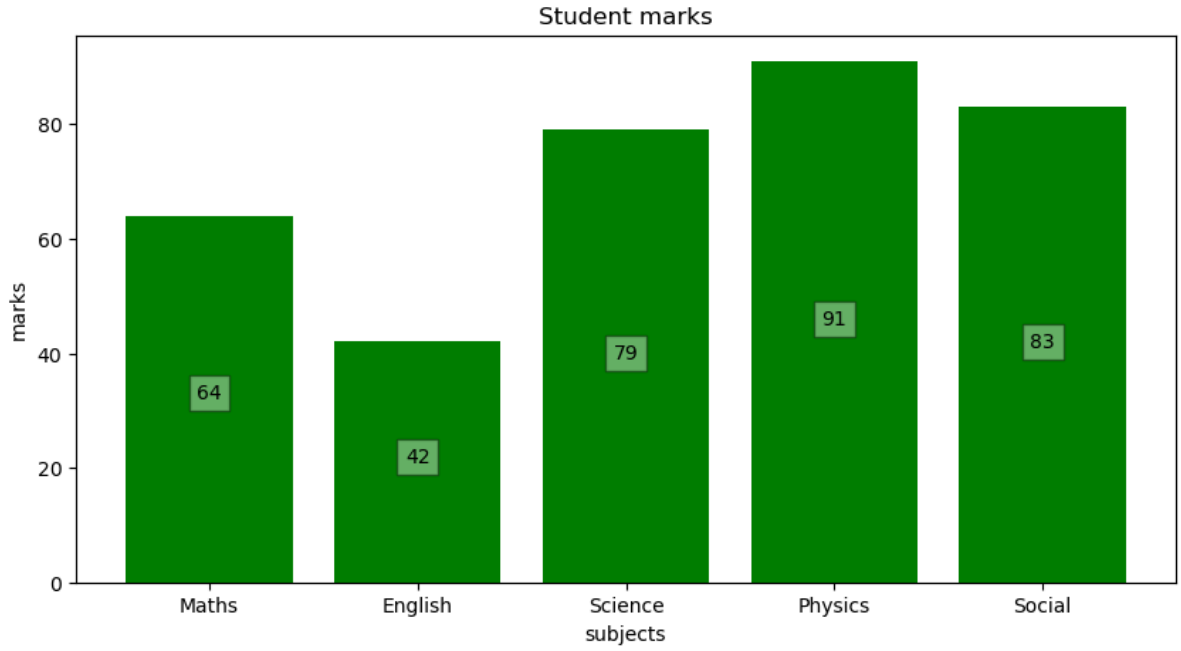
```
def add_labels(x, y):
    for i in range(len(x)):
        plt.text(i, y[i], y[i], ha='center', bbox=dict(facecolor='red',
alpha=0.8))
x = ['Maths', 'English', 'Science', 'Physics', 'Social']
y = [64, 42, 79, 91, 83]
plt.figure(figsize=(10, 5))
plt.bar(x, y)
add_labels(x, y)
plt.title("Student marks")
plt.xlabel("subjects")
```

```
plt.ylabel("marks")
plt.show()
```



In [12]:

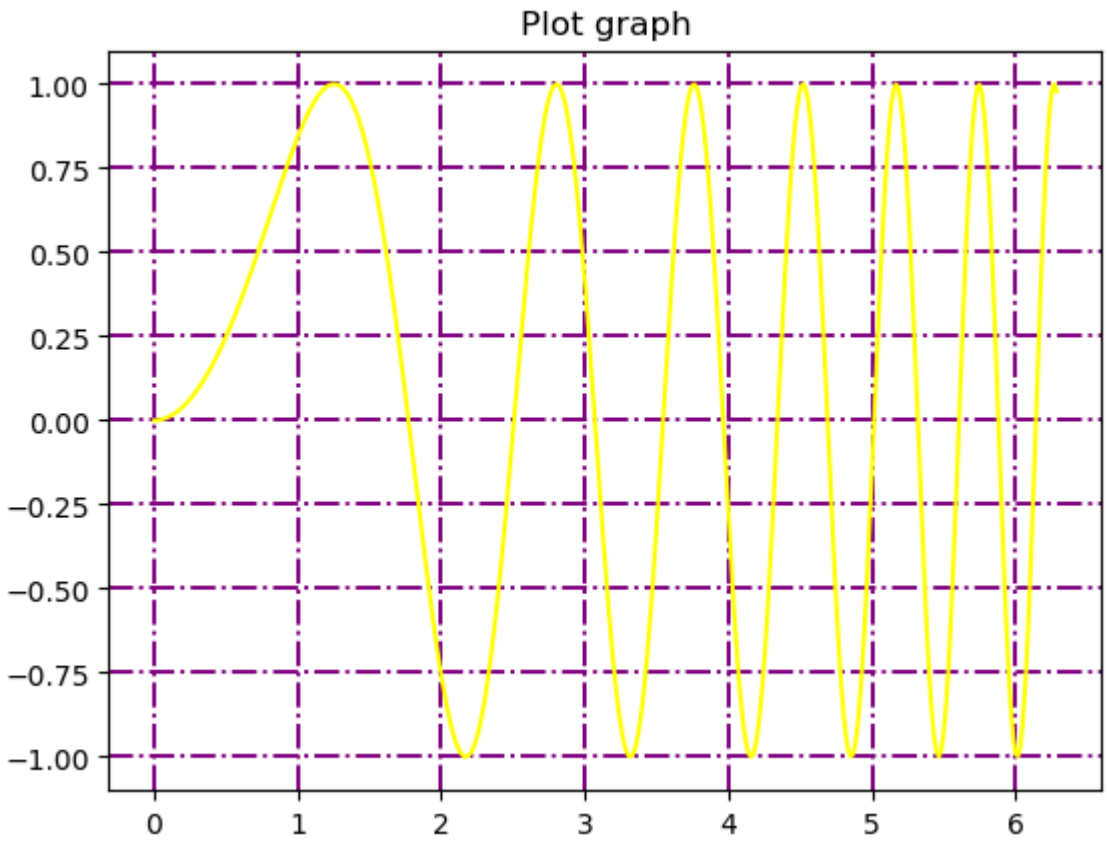
```
#Centered Labels Inside a Rectangular Box
def add_labels(x, y):
    for i in range(len(x)):
        plt.text(i, y[i] // 2, y[i], ha='center',
bbox=dict(facecolor='w', alpha=0.4))
x = ['Maths', 'English', 'Science', 'Physics', 'Social']
y = [64, 42, 79, 91, 83]
plt.figure(figsize=(10, 5))
plt.bar(x, y, color='green')
add_labels(x, y)
plt.title("Student marks")
plt.xlabel("subjects")
plt.ylabel("marks")
plt.show()
```



Grids in Matplotlib

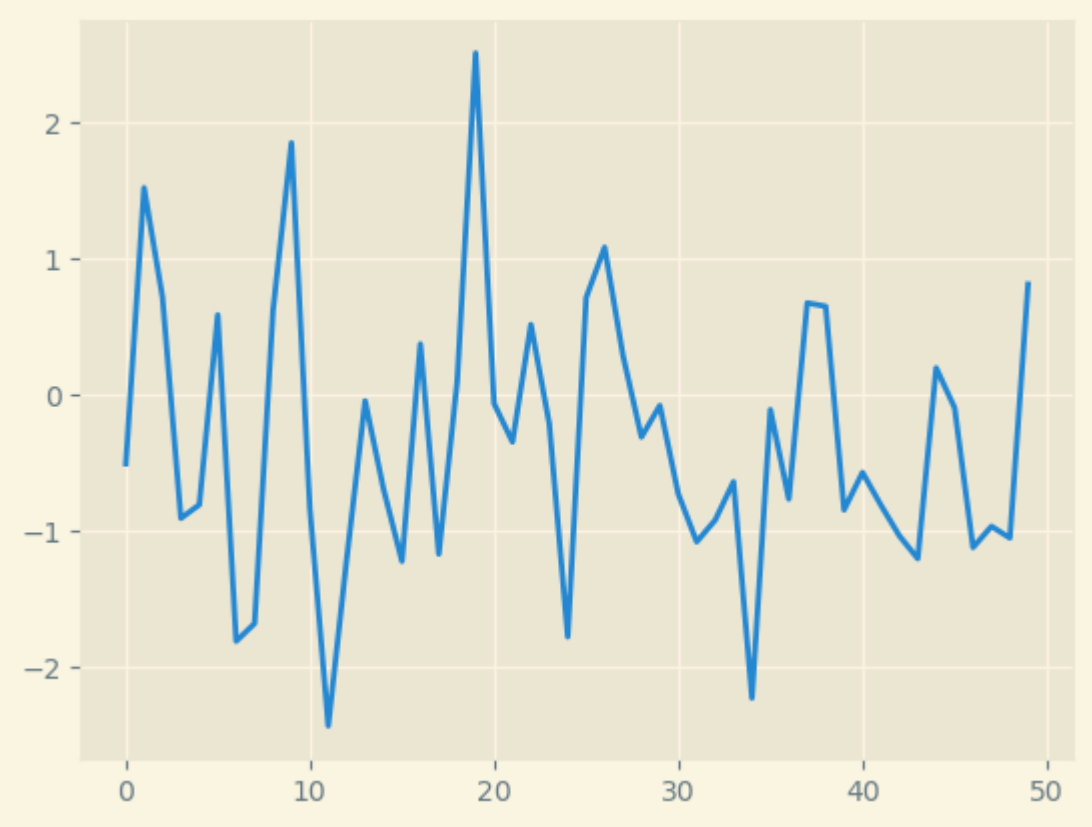
Syntax: matplotlib.pyplot.grid(True, color = "grey", linewidth = "1.4", axis = ''Y", linestyle = "-.") Parameters: True/False: Specifies whether the grid should be displayed. color: Defines the color of the grid lines. linewidth: Sets the thickness of the grid lines. axis: Determines which axis to apply the grid ("x", "y", or both). linestyle: Specifies the style of the grid lines (e.g., dashed, dotted, etc.).

In [16]:
```
x = np.linspace(0, 2 * np.pi, 400)
y = np.sin(x ** 2)
plt.plot(x, y, color='yellow')
plt.title("Plot graph")
plt.grid(True, color='purple', linewidth='1.4', linestyle='-.')
```



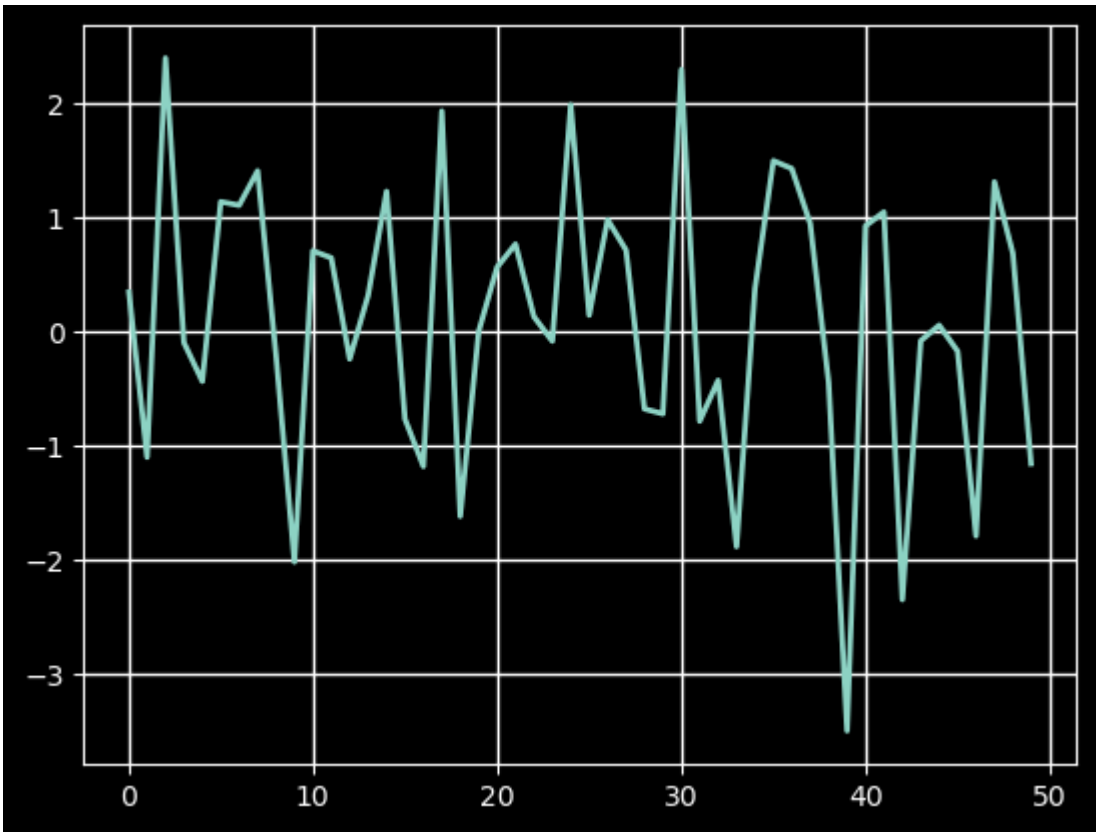
Style Plots using Matplotlib

In [17]:
```
from matplotlib import style
data = np.random.randn(50)
plt.style.use('Solarize_Light2')
plt.plot(data)
plt.show()
```



In [18]:

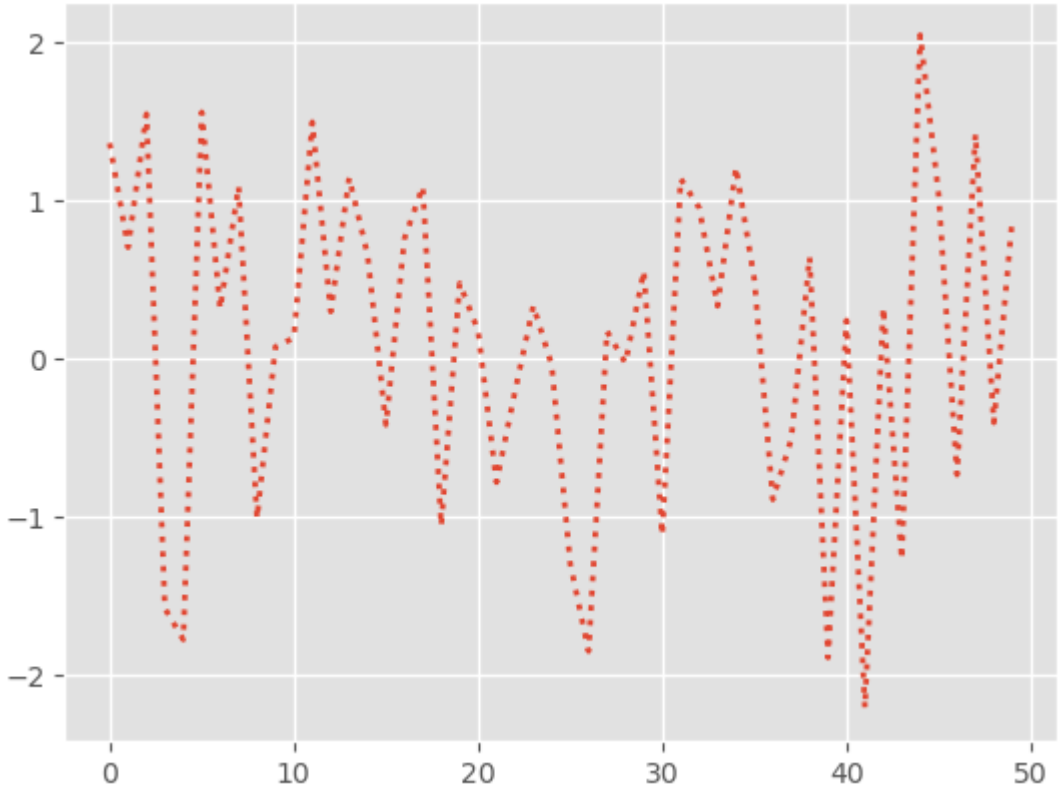
```
data = np.random.randn(50)
plt.style.use('dark_background')
plt.plot(data)
plt.show()
```



In [19]:

```
data = np.random.randn(50)
plt.style.use('ggplot')
```

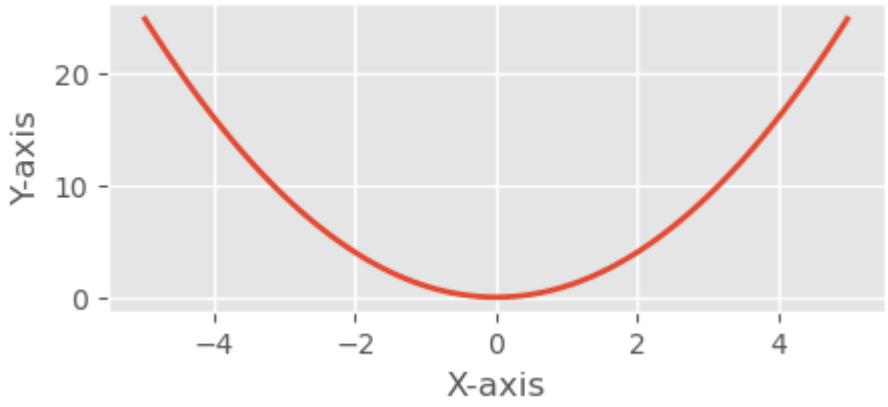
```
plt.plot(data, linestyle=":", linewidth=2)
plt.show()
```



Changing te size of figures

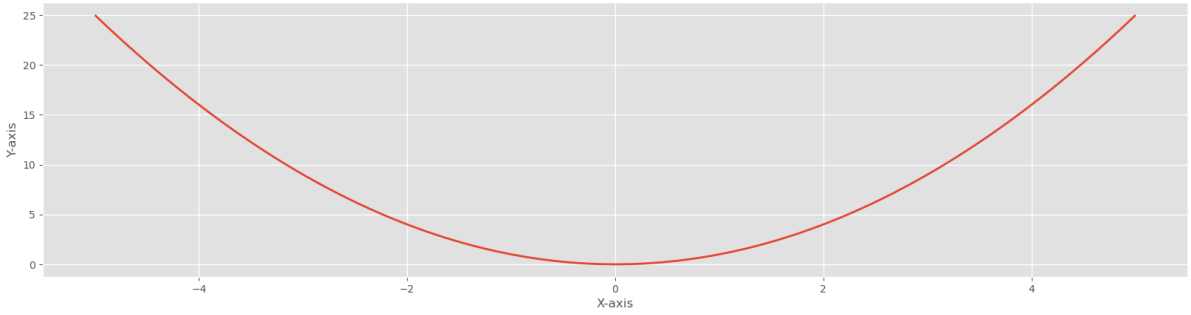
In [24]:

```
x = np.linspace(-5, 5, 100)
y = x**2
plt.figure(figsize=(5, 2))
plt.plot(x, y)

plt.title('Plot of y = x^2')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```



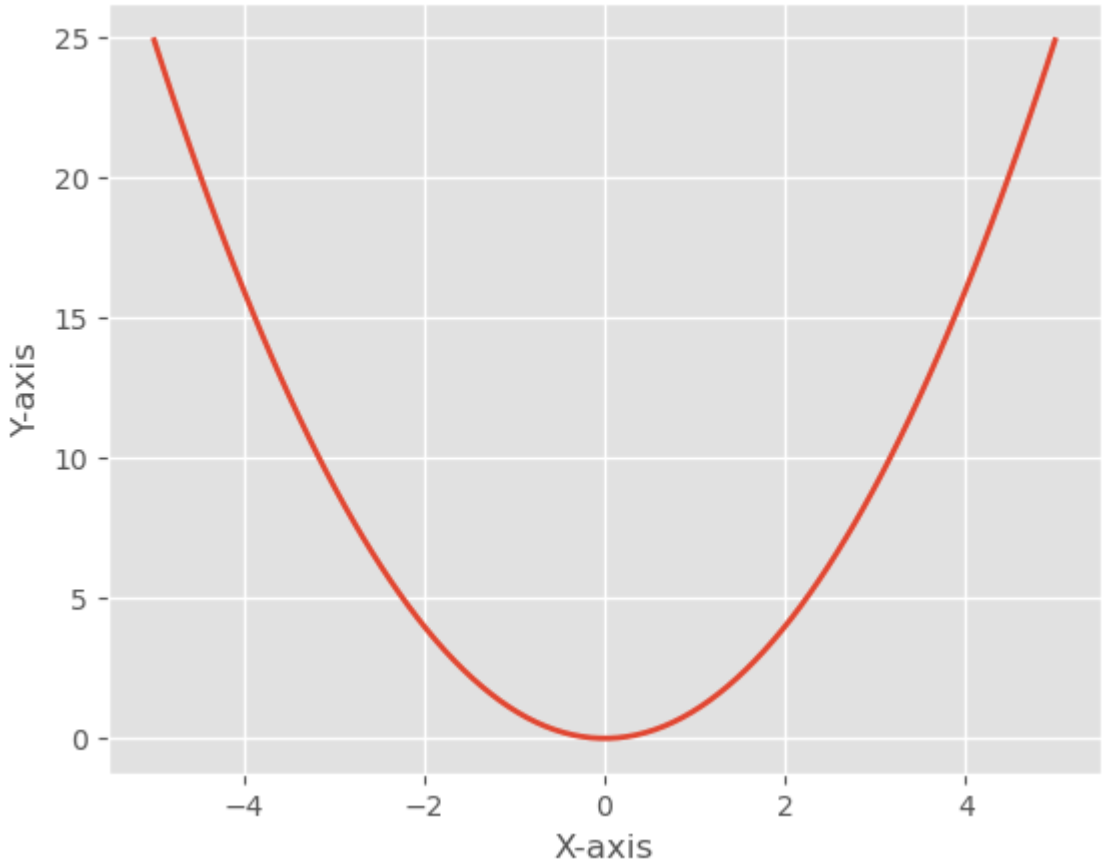
In [25]:

```
x = np.linspace(-5, 5, 100)
y = x**2
plt.figure().set_figwidth(20)
plt.plot(x, y)

plt.title('Plot of y = x^2')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```



In [26]:

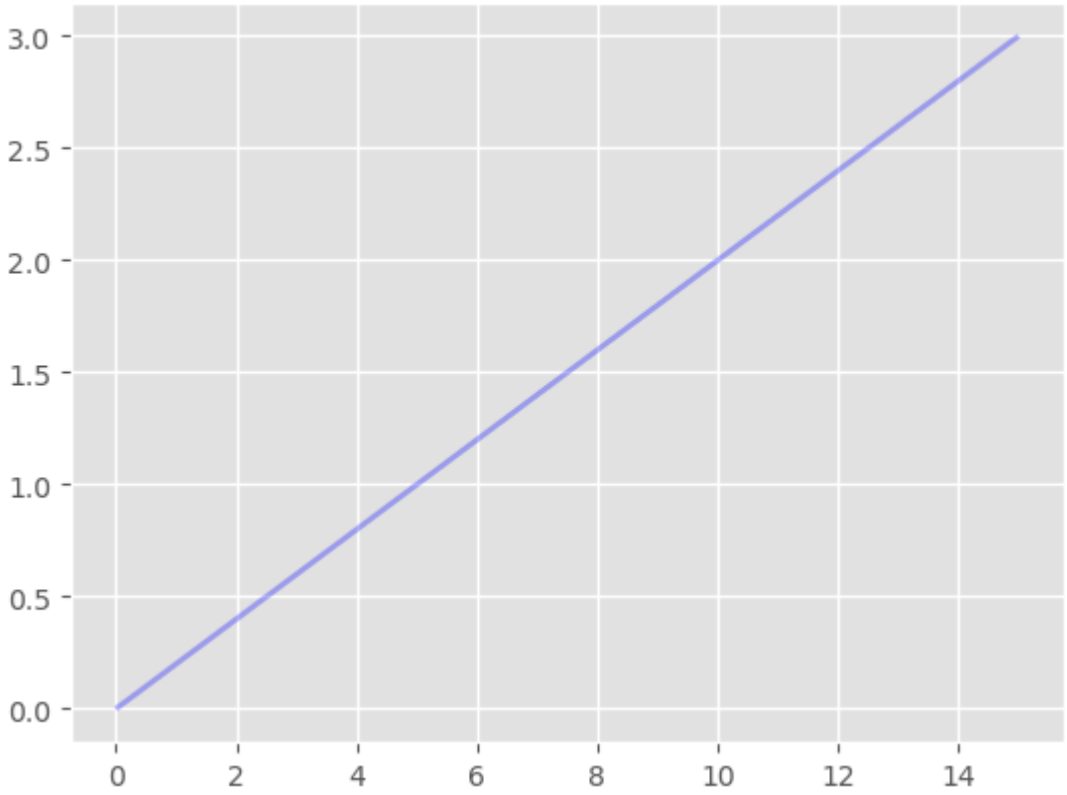
```
x = np.linspace(-5, 5, 100)
y = x**2
plt.figure().set_figheight(5)
plt.plot(x, y)

plt.title('Plot of y = x^2')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```



Transparency of Graph Plot

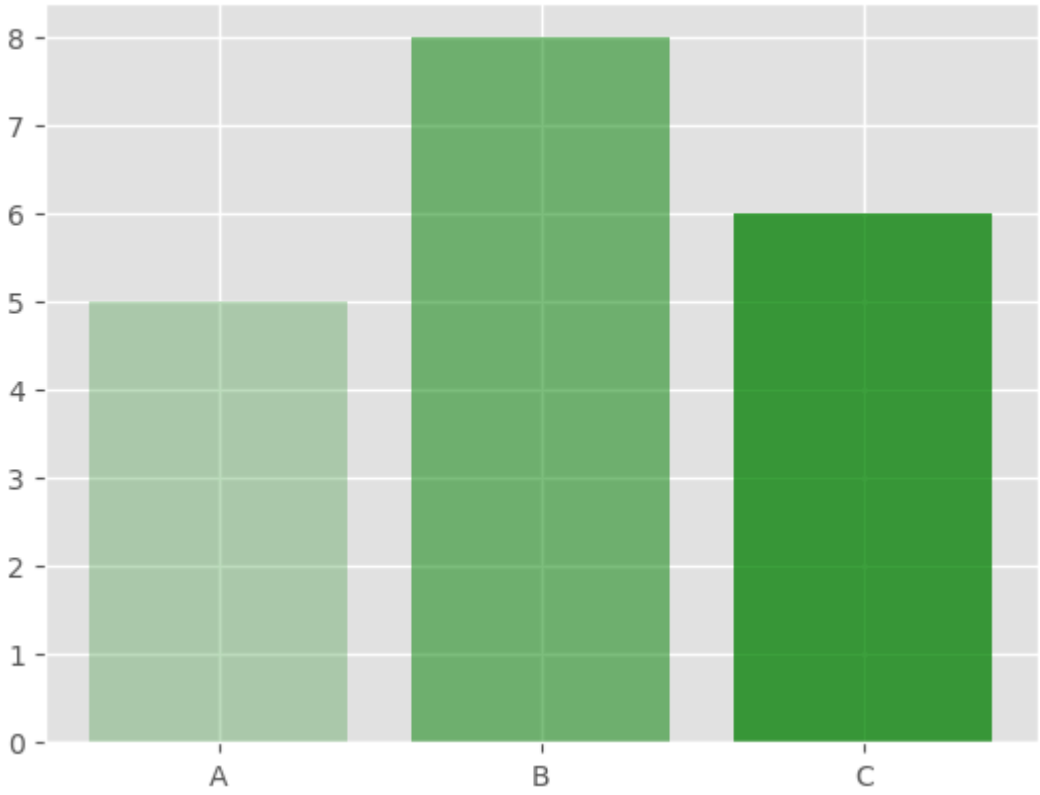
In [ ]:
```
alpha parameter is the most direct way to control transparency when
creating plots in Matplotlib.
```

In [27]:
```
x = [0, 5, 10, 15]
y = [0, 1, 2, 3]

plt.plot(x, y, color='blue', alpha=0.3)
plt.title("Line plot with alpha=0.3")
plt.show()
```



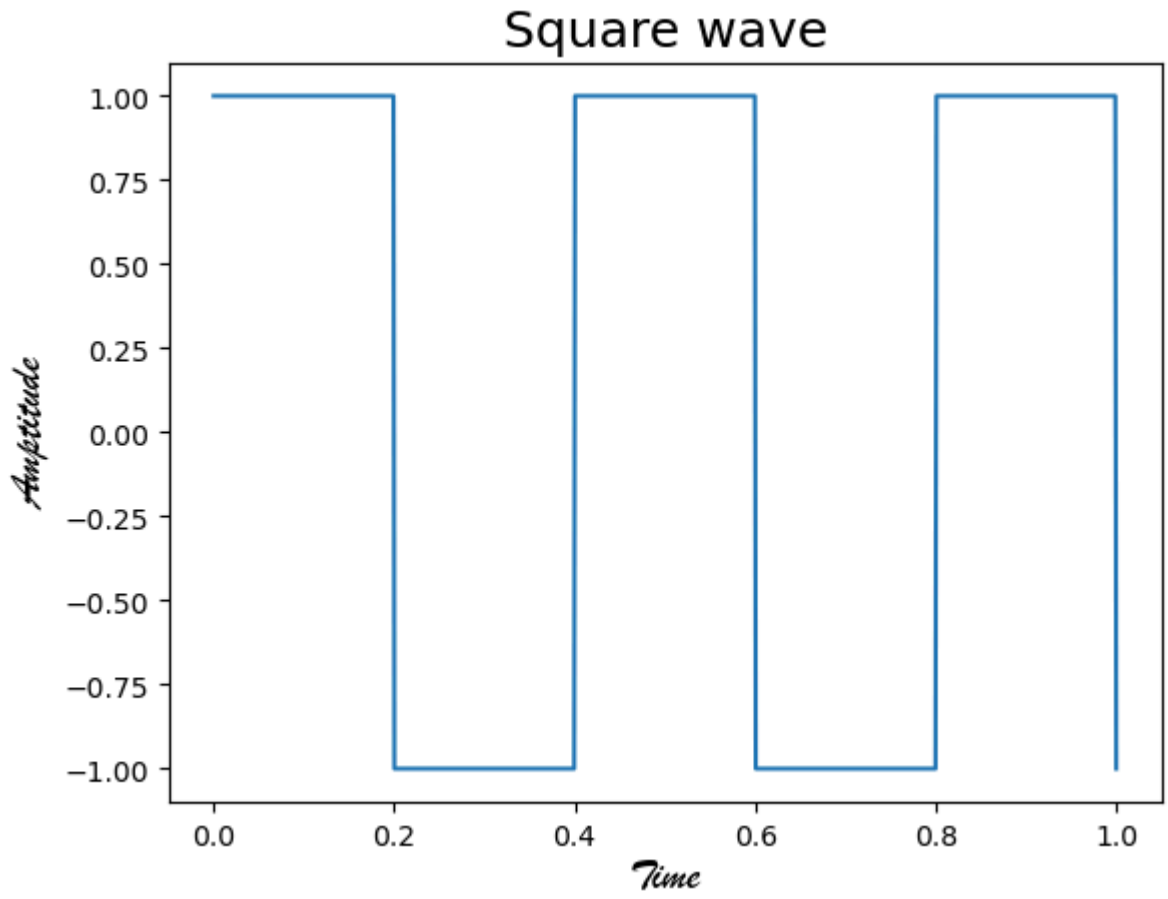
In [28]:

```
x = ['A', 'B', 'C']
y = [5, 8, 6]

for i in range(len(x)):
    plt.bar(x[i], y[i], color='green', alpha=(i + 1)/4)

plt.title("Bar chart with varying alpha")
plt.show()
```



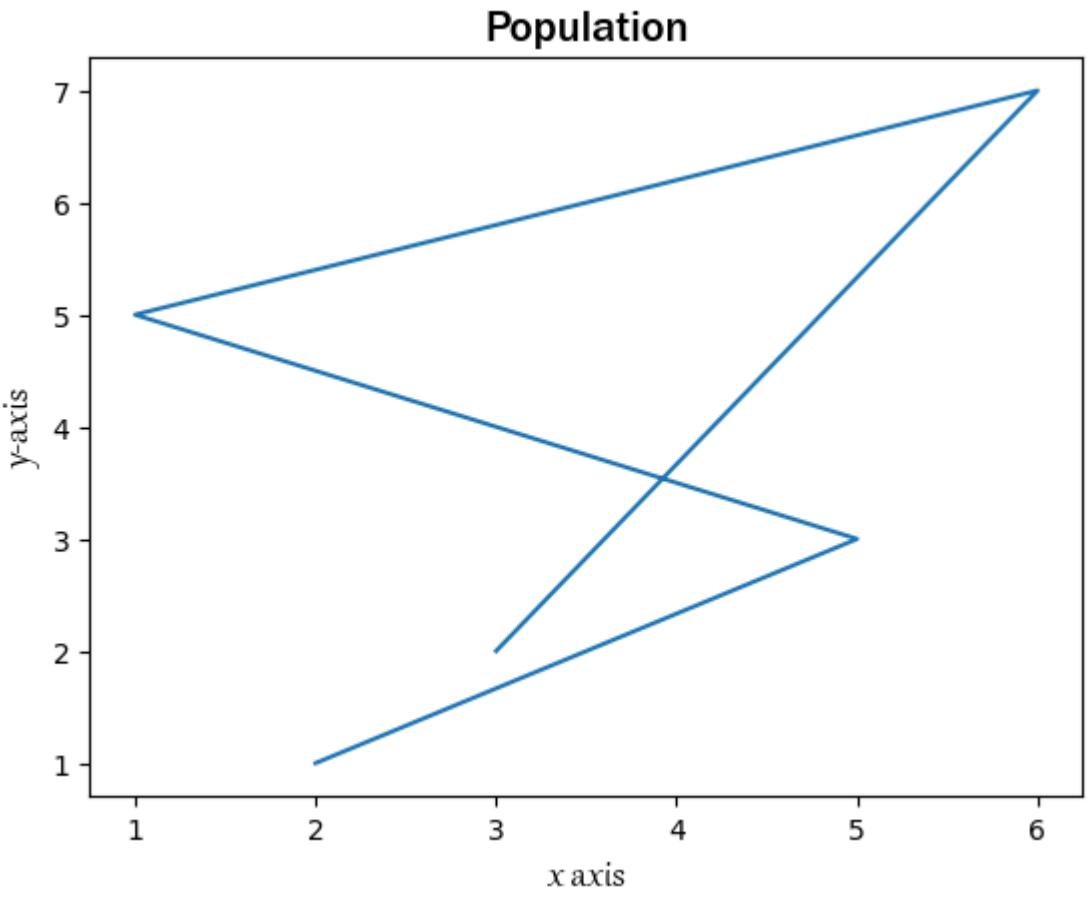
Changing fonts in Matplotlib

In [3]:
```
#using fontname
from scipy import signal
t = np.linspace(0, 1, 1000, endpoint=True)
plt.plot(t, signal.square(np.pi*5*t))
plt.xlabel("Time", fontname = "Brush Script Mt", fontsize=16)
plt.ylabel("Amptitude", fontname="Brush Script MT", fontsize=16)
plt.title("Square wave", fontsize=18)
plt.show()
```



In [6]:

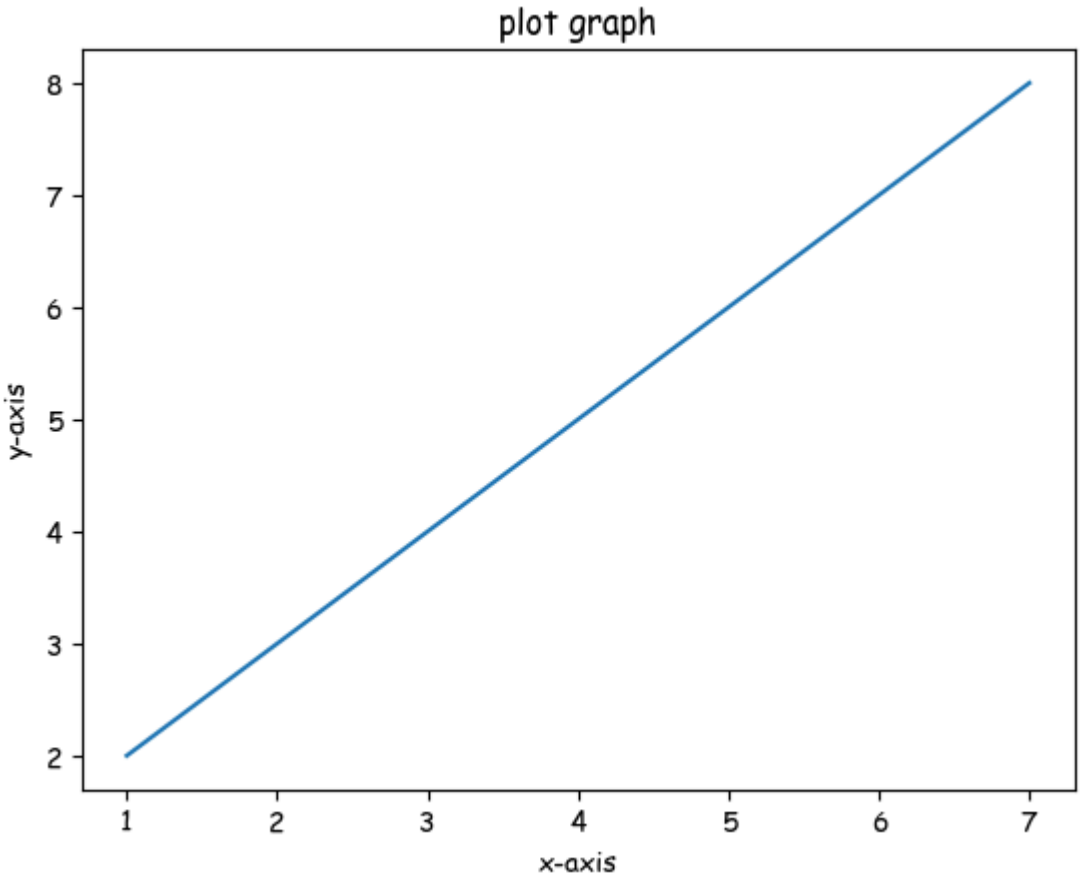
```
x = [2, 5, 1, 6, 3]
y = [1, 3, 5, 7, 2]
plt.plot(x, y)
plt.title("Population", fontname="Franklin Gothic Medium", fontsize=16)
plt.xlabel("x axis", fontname="Gabriola", fontsize=16)
plt.ylabel("y-axis", fontname="Gabriola", fontsize=16)
plt.show()
```



using rcParams

In [7]:

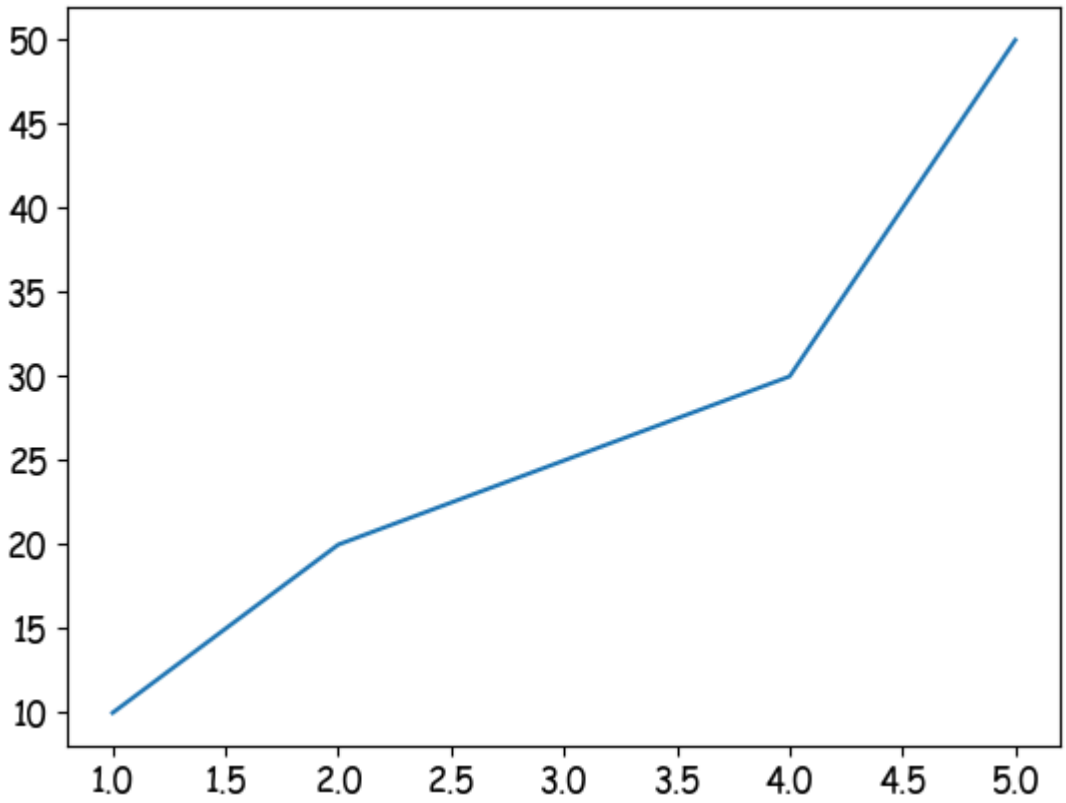
```
#used if we need to apply font style to all elemnts like title, labels
and ticks
import matplotlib as mp1
mp1.rcParams['font.family'] = "Comic Sans MS"
x = [1, 3, 5, 7]
y = [2, 4, 6, 8]
plt.plot(x, y)
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("plot graph")
plt.show()
```



Setting Tick Labels Font Size in Matplotlib

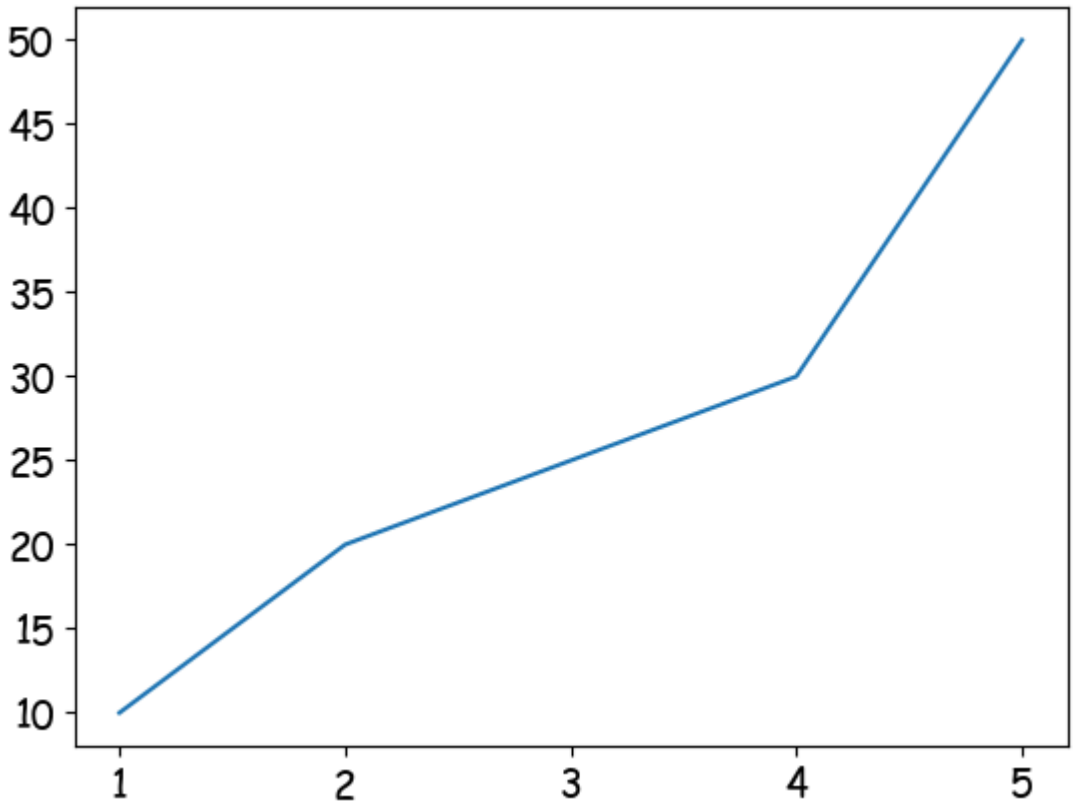
1. using font size
2. using set_tick_params()
3. using set_fontsize()
4. using rcParams

In [8]:
```
#using fontsize
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 50]
plt.plot(x, y)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```

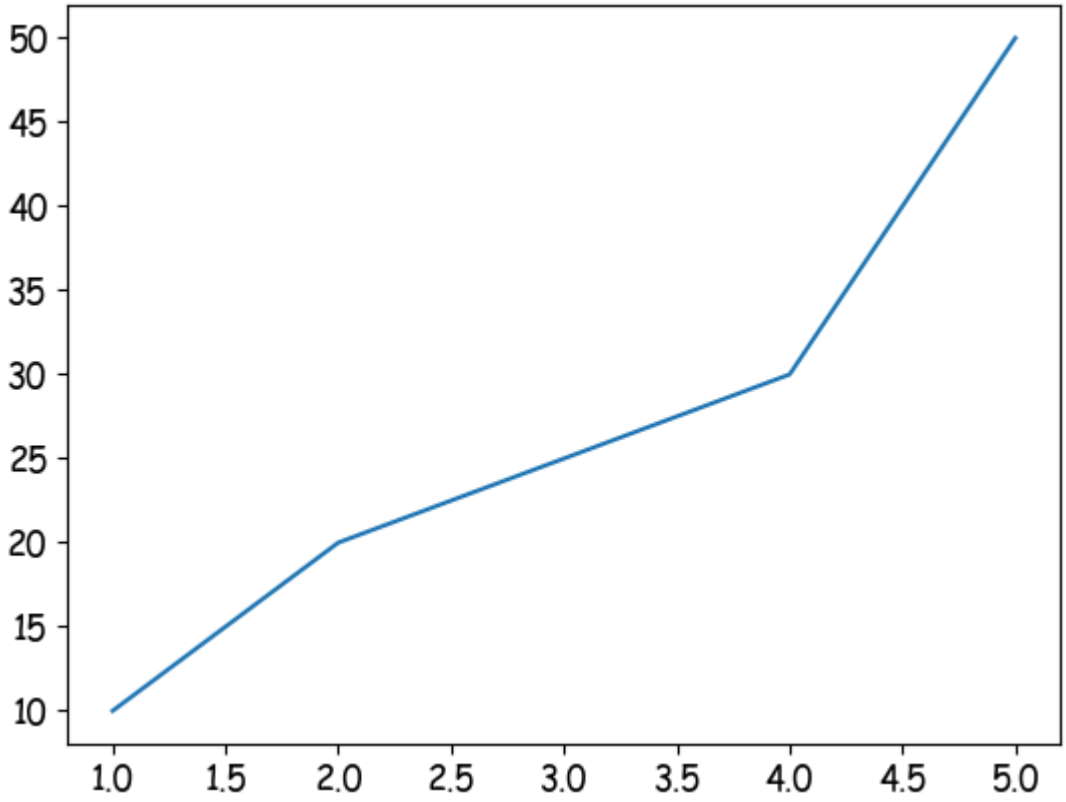


In [11]:

```
#using set_tick_params
fig, ax = plt.subplots()
ax.plot(x, y)
ax.tick_params(axis = 'both', which = 'major', labelsize=14)
ax.tick_params(axis = 'both', which = 'minor', labelsize=10)
plt.show()
```

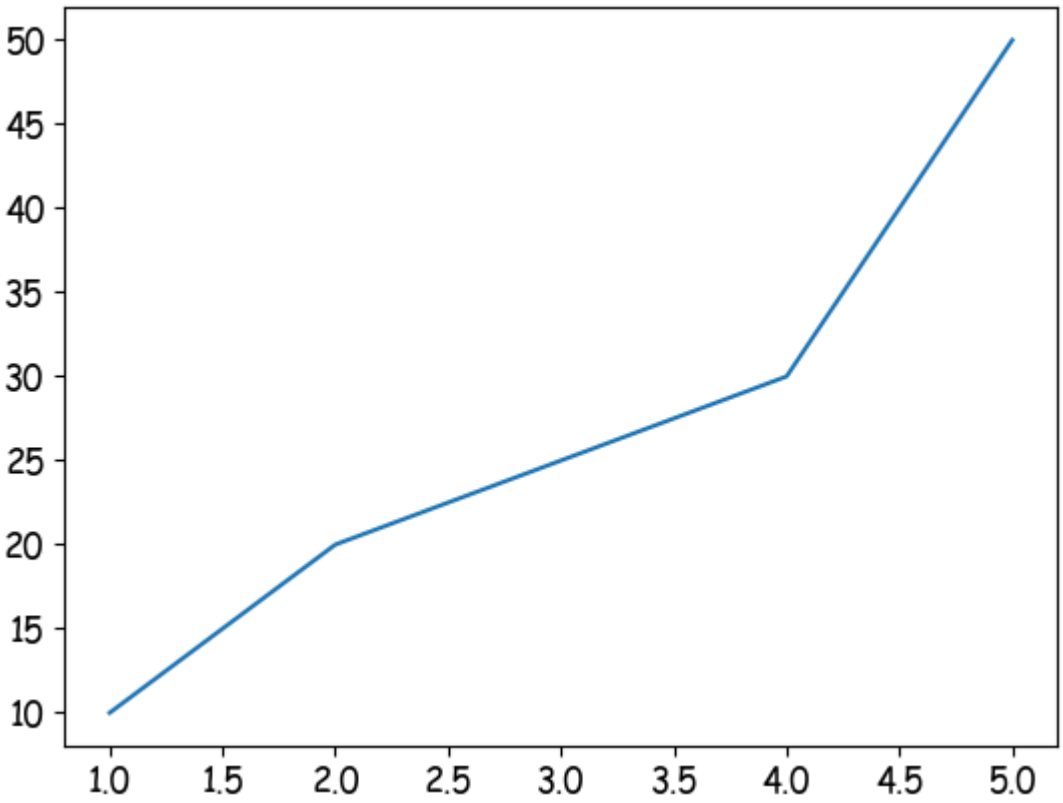


In [12]:
```
#using set_fontsize()
fig, ax = plt.subplots()
ax.plot(x, y)

# Modify tick label font sizes individually
for label in ax.get_xticklabels():
    label.set_fontsize(12)

for label in ax.get_yticklabels():
    label.set_fontsize(12)

plt.show()
```



using rcParams

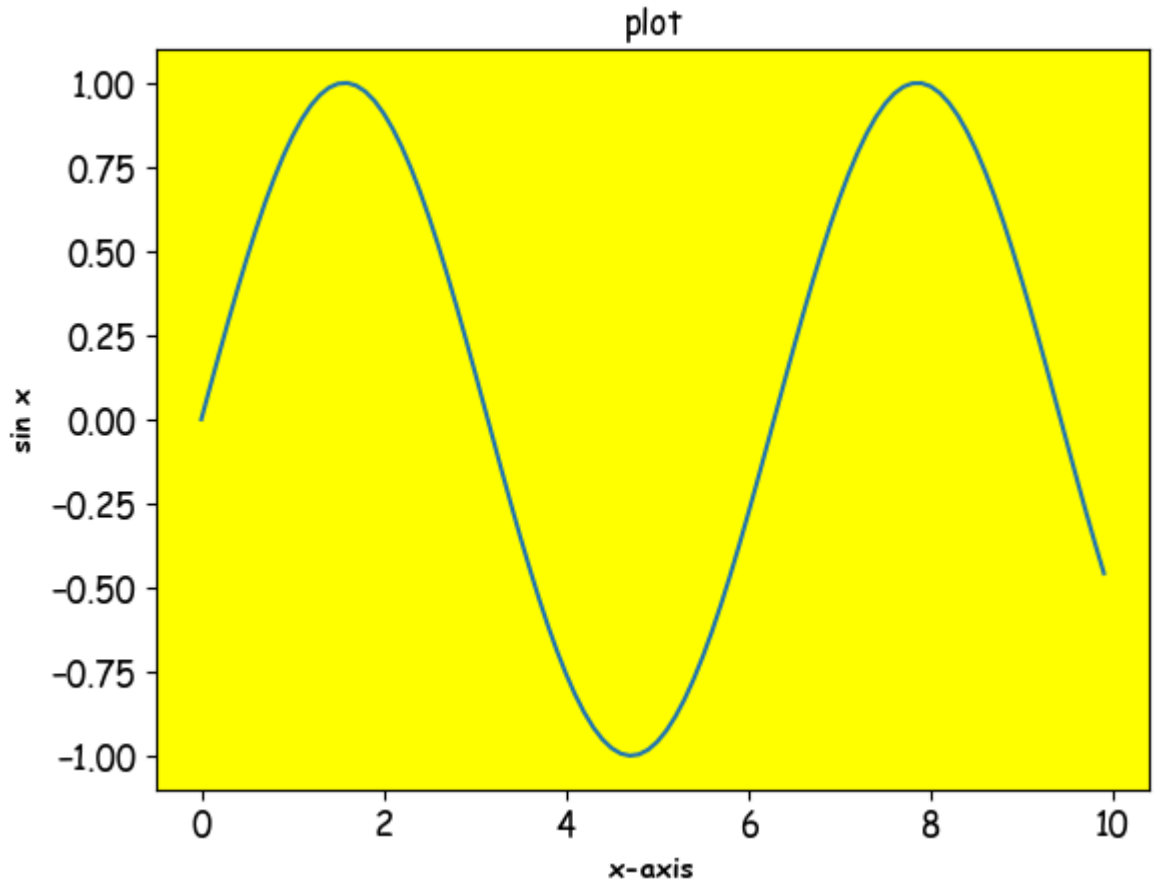
In [14]:
```
import matplotlib as mc1
mc1.rcParams['xtick.labelsize'] = 12
mc1.rcParams['ytick.labelsize'] = 12
fig, ax = plt.subplots()
ax.plot(x, y)
plt.show()
```



Setting Backgorund colour of plot in Matplotlib

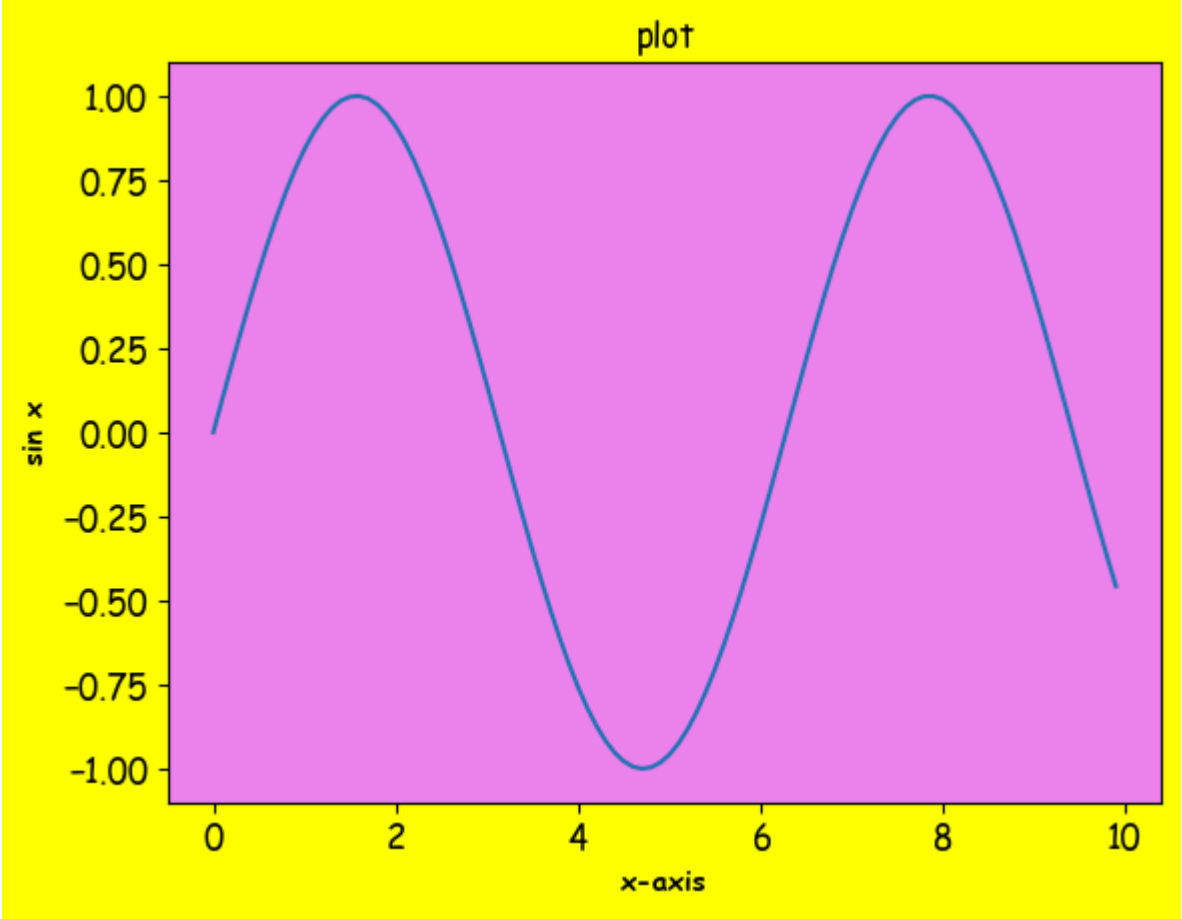
In [19]:

```
x = np.arange(0, 10, 0.1)
y = np.sin(x)
plt.plot(x, y)
plt.xlabel("x-axis", fontweight='bold')
plt.ylabel("sin x", fontweight='bold')
plt.title("plot")
ax = plt.gca()
ax.set_facecolor("yellow")
plt.show()
```



In [23]:

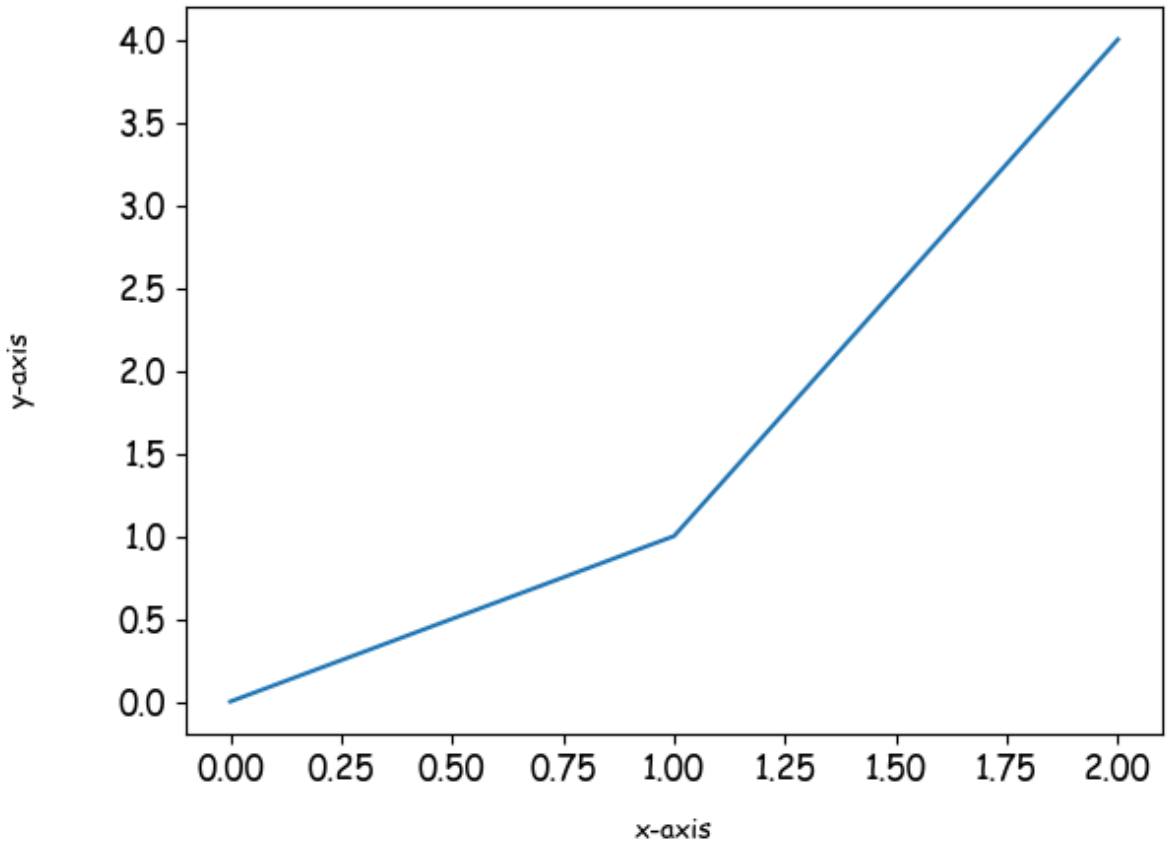
```
#Setting color inner and outer
x = np.arange(0, 10, 0.1)
y = np.sin(x)
plt.figure(facecolor='yellow')
plt.plot(x, y)
plt.xlabel("x-axis", fontweight='bold')
plt.title("plot")
ax = plt.gca()
ax.set_facecolor("violet")
plt.ylabel("sin x", fontweight="bold")
plt.show()
```



Adjusting the position of axis labels in Matplotlib

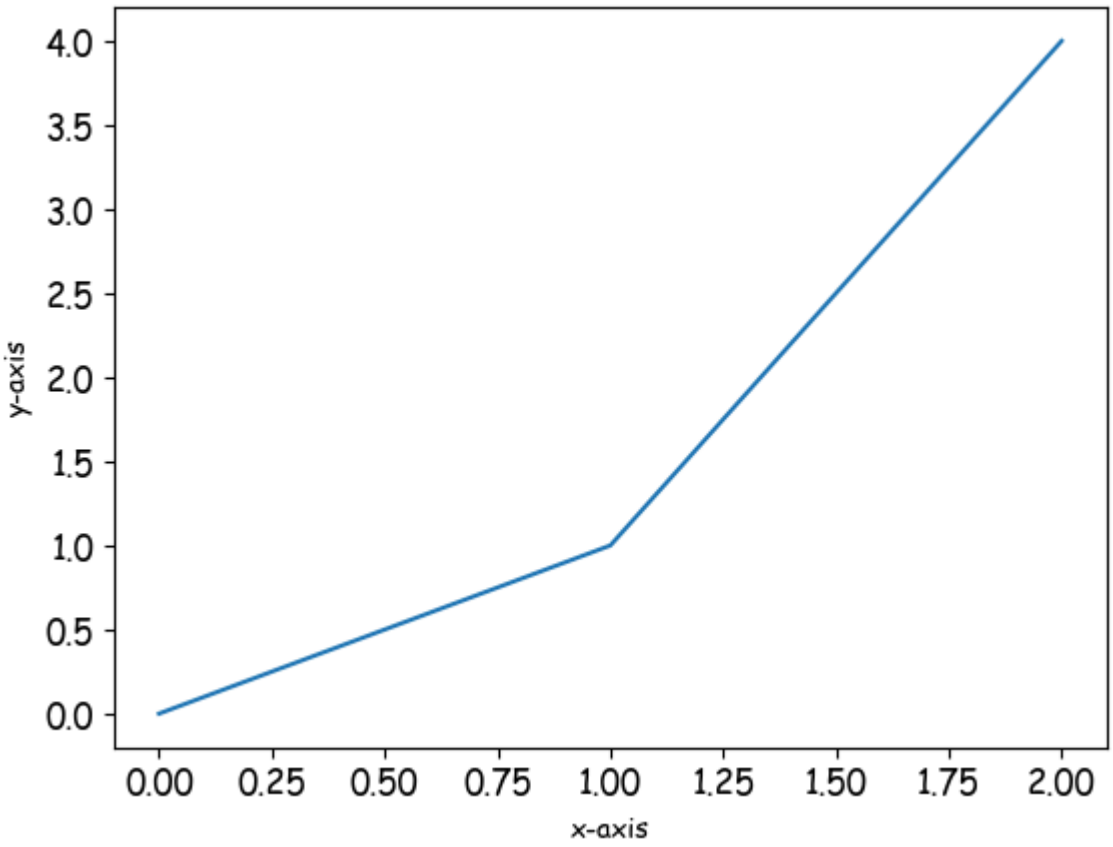
1.using set_xlabel() and set_ylabel() 2.using set_positon() 3.using set_rotation()

In [26]:
```
#using set_xlabel() and set_ylabel()
fig, ax = plt.subplots()
ax.plot([0,1,2], [0,1,4])
ax.set_xlabel("x-axis", labelpad=10)
ax.set_ylabel("y-axis", labelpad=30)
plt.show()
```



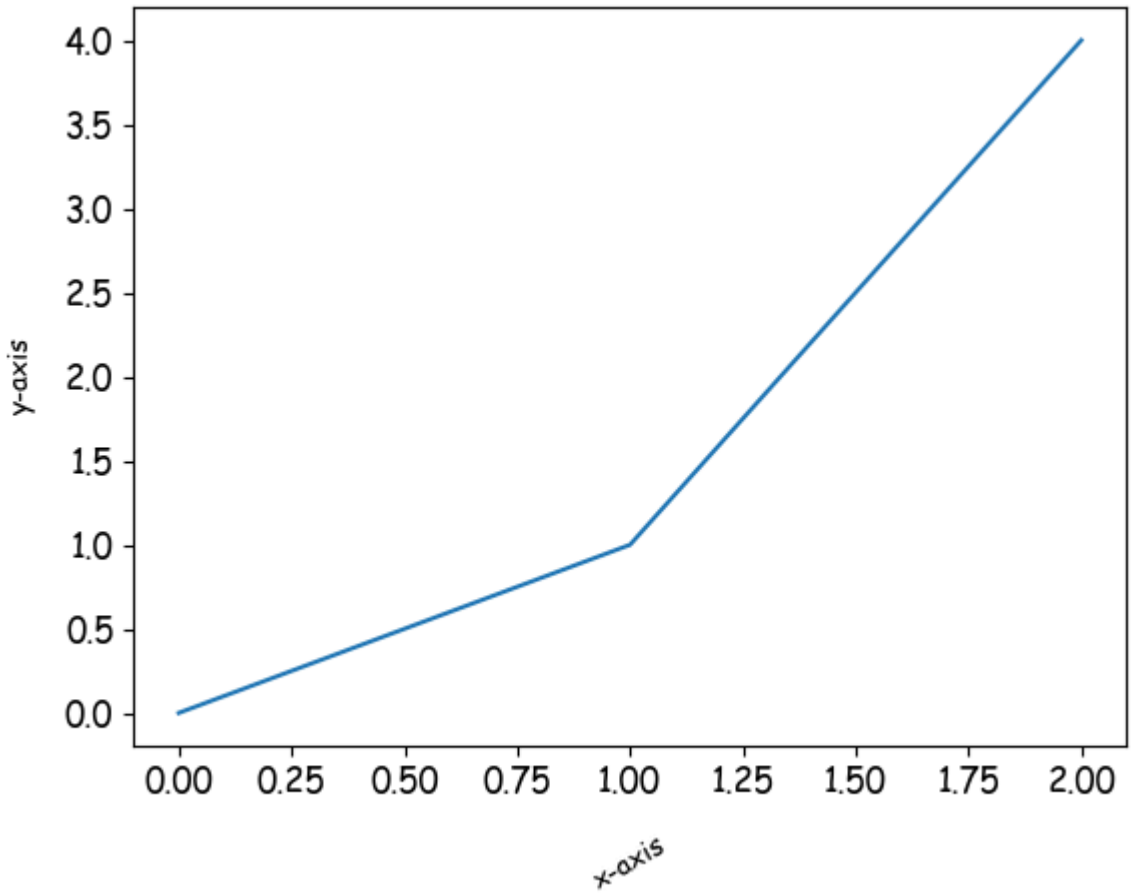
In [29]:

```
#usig set_position()
fig, ax = plt.subplots()
ax.plot([0,1,2], [0,1,4])
ax.xaxis.label.set_position((0.1, 0.5))
ax.yaxis.label.set_position([-0.2, 0.5])
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.show()
```



In [33]:

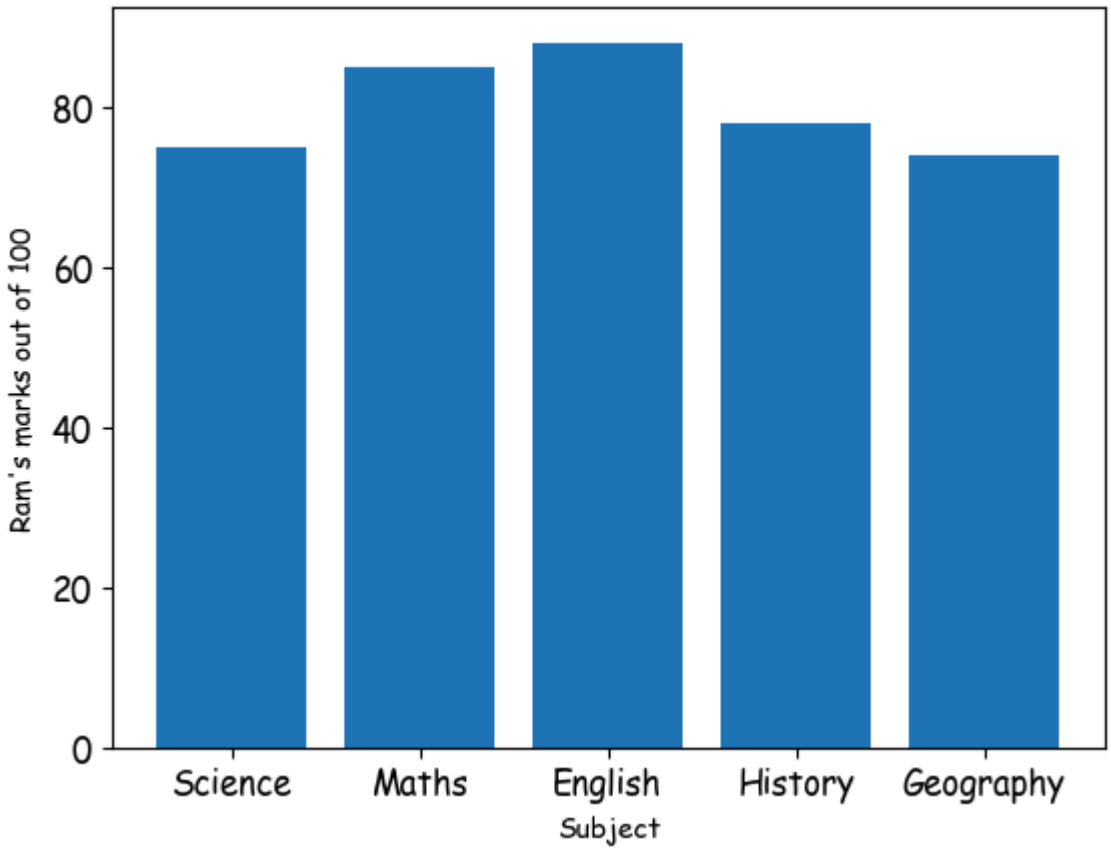
```
#using set_rotation()
fig, ax = plt.subplots()
ax.plot([0,1,2], [0,1,4])
ax.set_xlabel("x-axis", labelpad=10)
ax.set_ylabel("y-axis", labelpad=10)
ax.xaxis.label.set_rotation(30)
ax.yaxis.label.set_rotation(90)
plt.show()
```



Hiding the axis, borders and whitespaces in Matplotlib

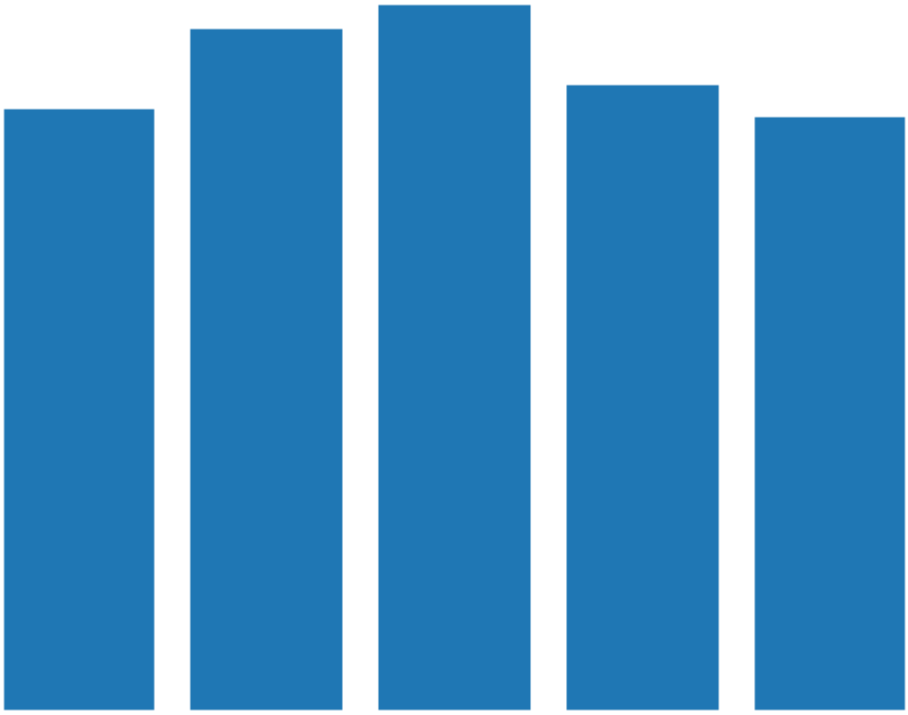
In [34]:

```
import numpy as np
import matplotlib.pyplot as plt

# Marks of RAM in different subjects out of 100
x = ['Science', 'Maths', 'English', 'History', 'Geography']
y = [75, 85, 88, 78, 74]

plt.bar(x, y)
plt.xlabel("Subject")
plt.ylabel("Ram's marks out of 100")
plt.show()
```



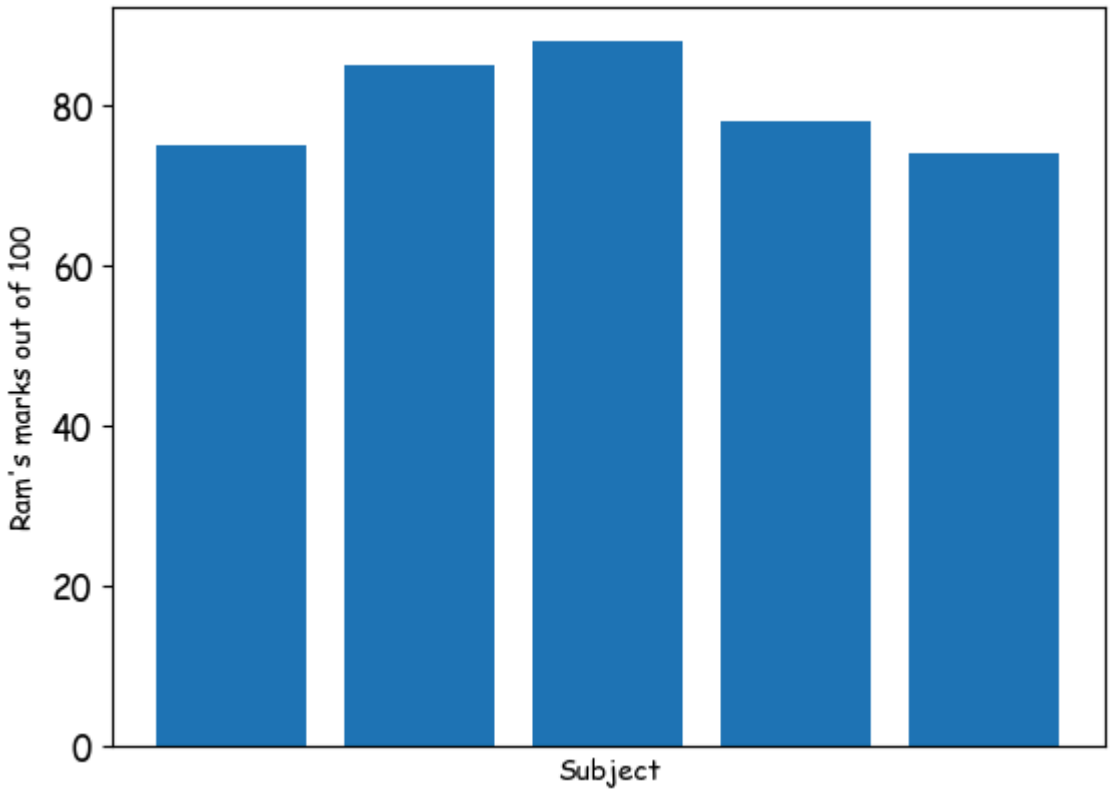
In [35]:

```
#Hiding axis on both sides
# Marks of RAM in different subjects out of 100
x = ['Science', 'Maths', 'English', 'History', 'Geography']
y = [75, 85, 88, 78, 74]

plt.bar(x, y)
plt.xlabel("Subject")
plt.ylabel("Ram's marks out of 100")
plt.axis("off")
plt.show()
```



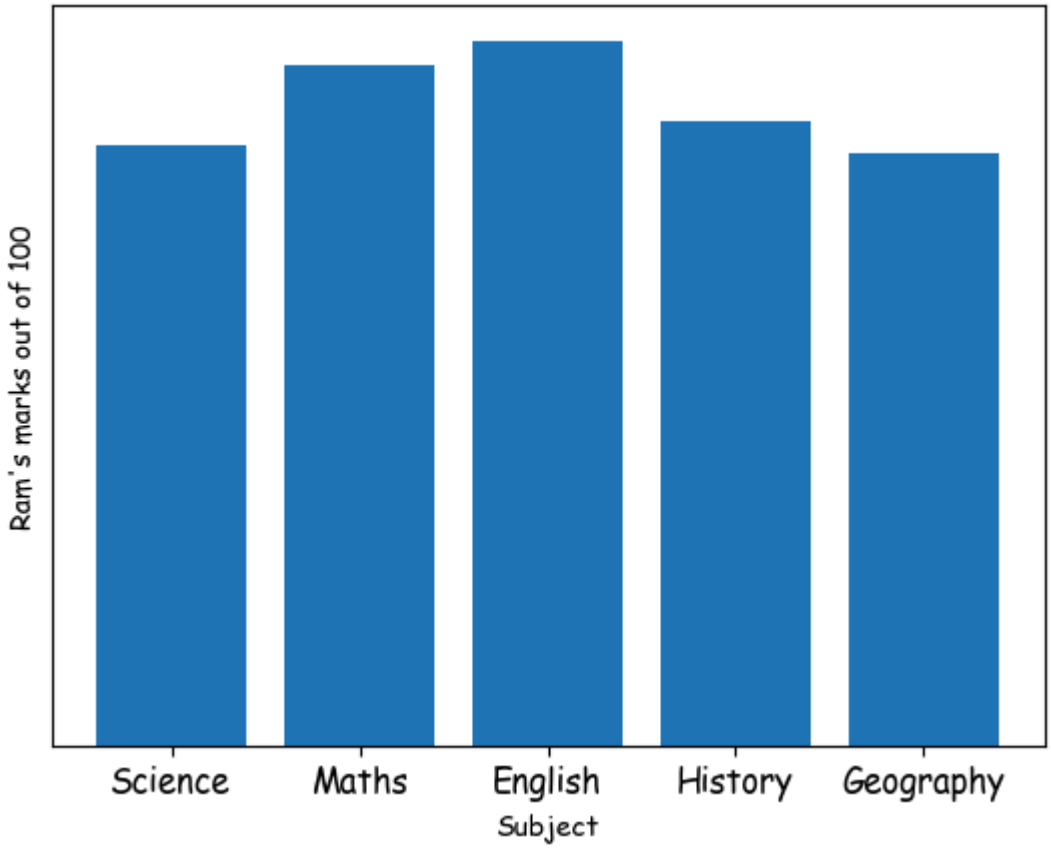
In [36]:

```
#Hiding x-axis
# Marks of RAM in different subjects out of 100
x = ['Science', 'Maths', 'English', 'History', 'Geography']
y = [75, 85, 88, 78, 74]

plt.bar(x, y)
plt.xlabel("Subject")
plt.ylabel("Ram's marks out of 100")
plt.xticks([])
plt.show()
```



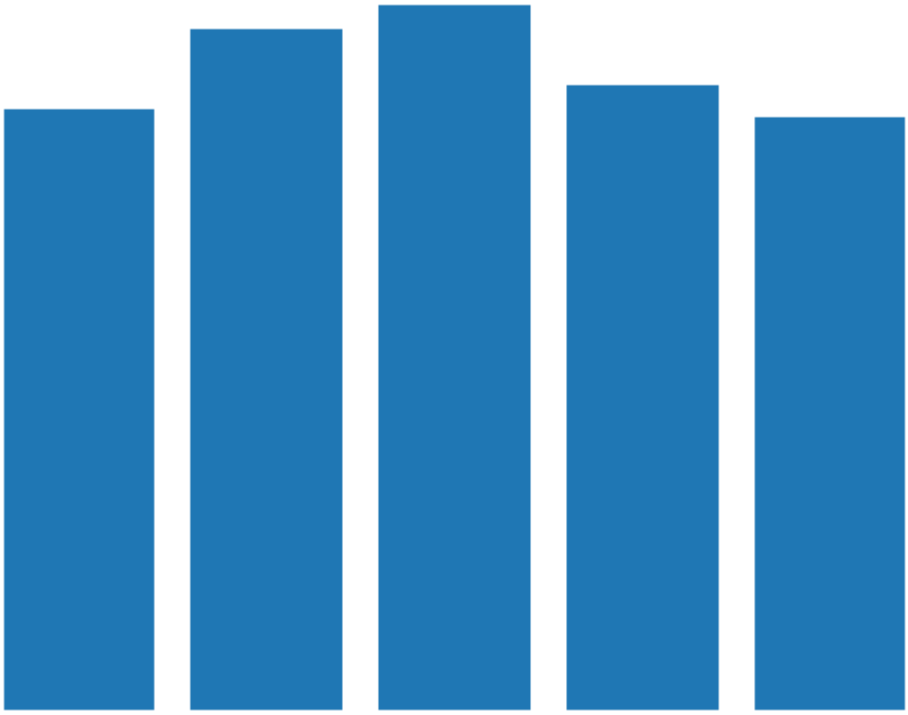
In [38]:

```
#Hiding y-axis
# Marks of RAM in different subjects out of 100
x = ['Science', 'Maths', 'English', 'History', 'Geography']
y = [75, 85, 88, 78, 74]

plt.bar(x, y)
plt.xlabel("Subject")
plt.ylabel("Ram's marks out of 100")
plt.yticks([])
plt.show()
```

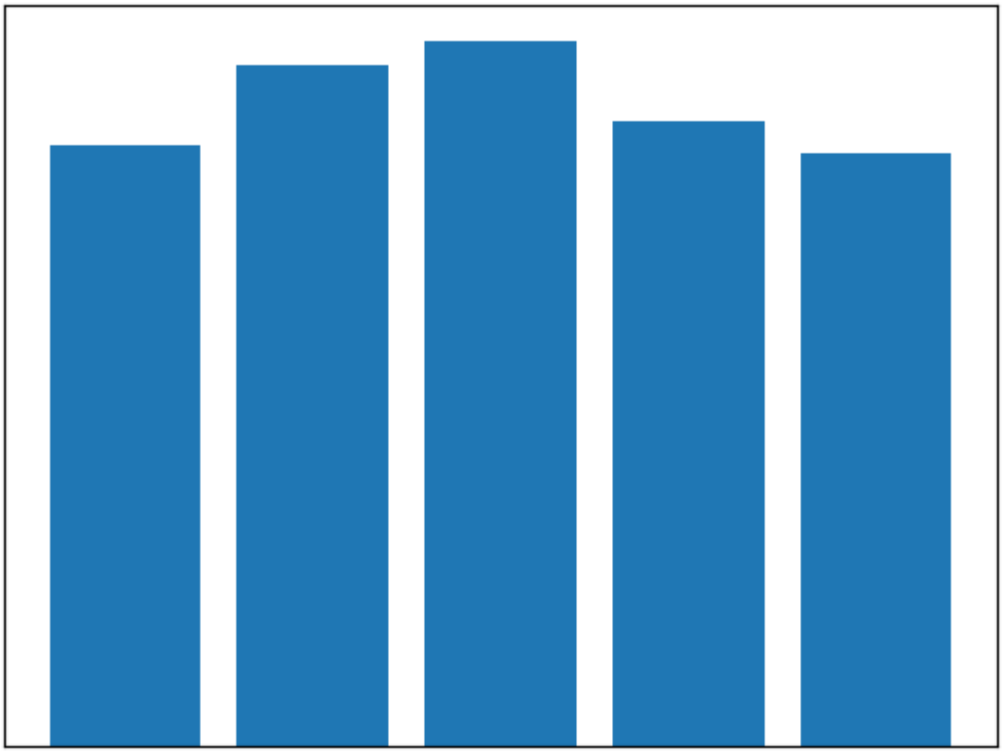


In [39]:

```
#Hiding borders and whitespaces
x = ['Science', 'Maths', 'English', 'History', 'Geography']
y = [75, 85, 88, 78, 74]

plt.bar(x, y)
plt.xlabel("Subject")
plt.ylabel("Ram's marks out of 100")
plt.axis('off')

# Save figure without borders and white spaces
plt.savefig('image.png', bbox_inches='tight', pad_inches=0)
plt.show()
```



In [40]:
```
#using ax.gca()
x = ['Science', 'Maths', 'English', 'History', 'Geography']
y = [75, 85, 88, 78, 74]

plt.bar(x, y)
plt.xlabel("Subject")
plt.ylabel("Ram's marks out of 100")

ax = plt.gca()
ax.xaxis.set_visible(False)  # Hide X-axis
ax.yaxis.set_visible(False)  # Hide Y-axis

plt.show()
```



In [ ]: