Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is widely used for data visualization and supports various types of plots such as line graphs, bar charts, scatter plots, and more.

Install Necessary Libararies

In [ ]:
```
! pip install matplotlib
```

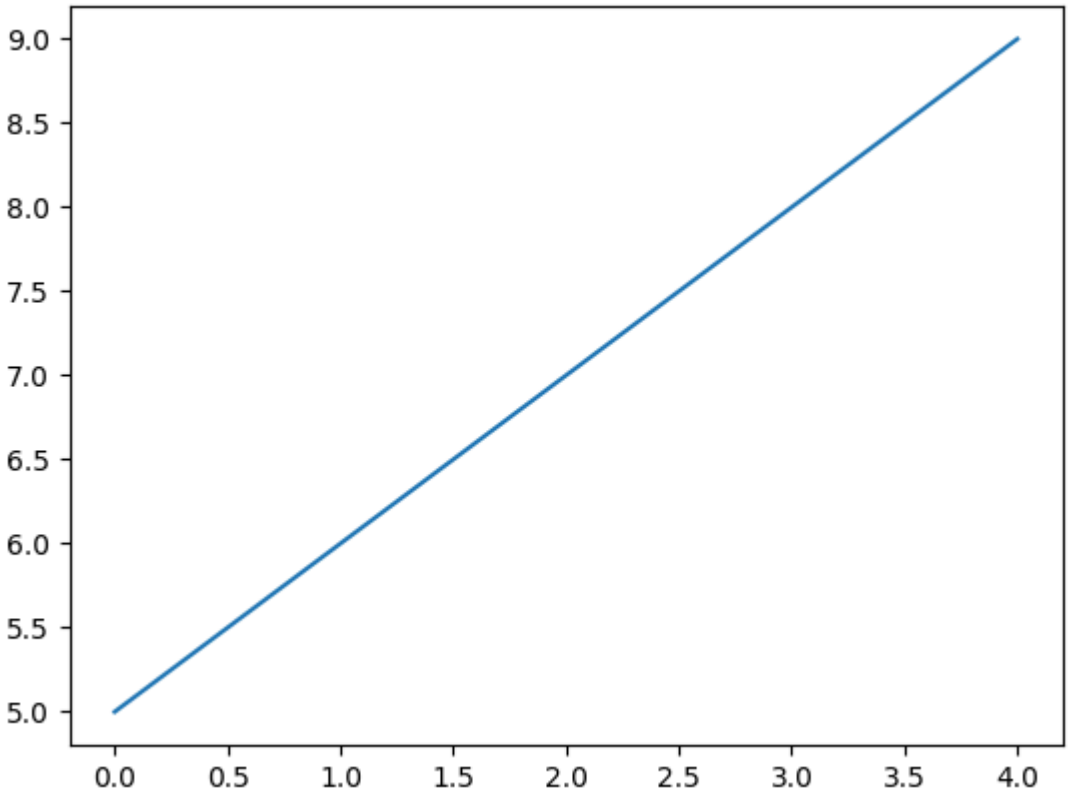
Load Necessary Libraries

In [2]:
```
import matplotlib.pyplot as plt # For plotting graphs
import numpy as np # For handling arrays
import pandas as pd # For handling csv files
```

Basic Graph (Line Graph)

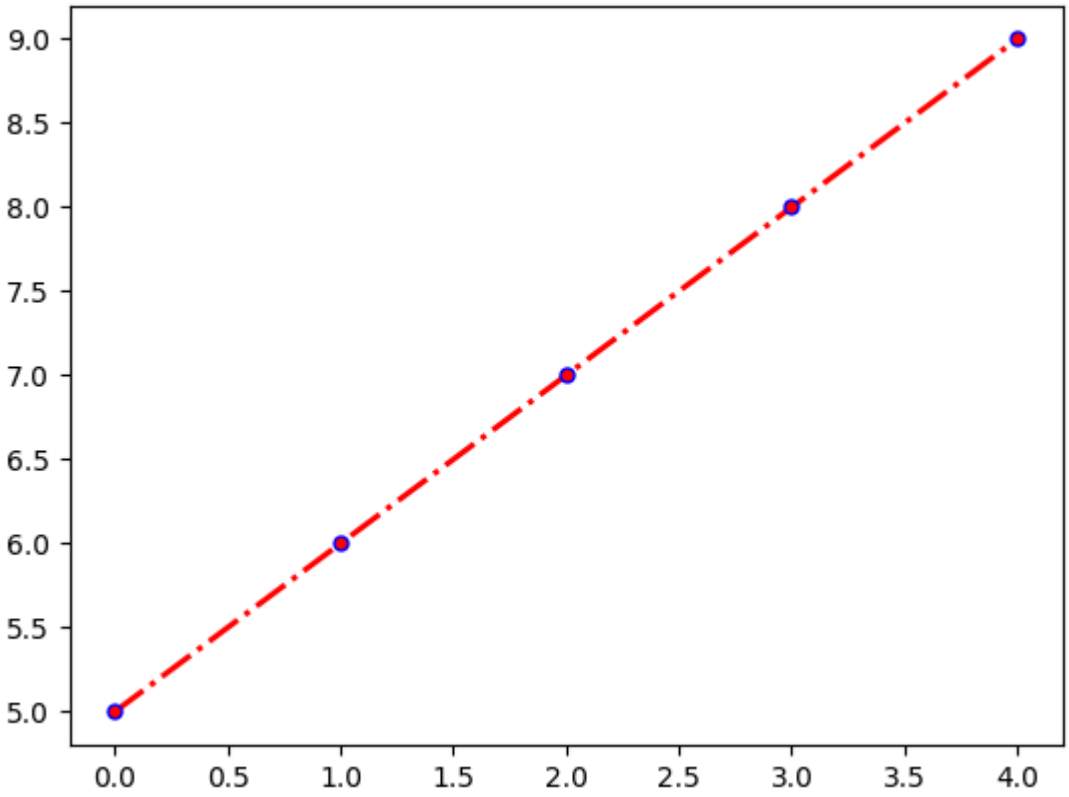
Basic Plot

In [3]:
```
# Define the values for (x, y) to make graph
x = [0, 1, 2, 3, 4] # Values for x
y = [5, 6, 7, 8, 9] # Values for y
# Remember that length of both array must be equal

# Plot x vs y graph
plt.plot(x, y) # plot takes 1D array

plt.show() # Show graph
```



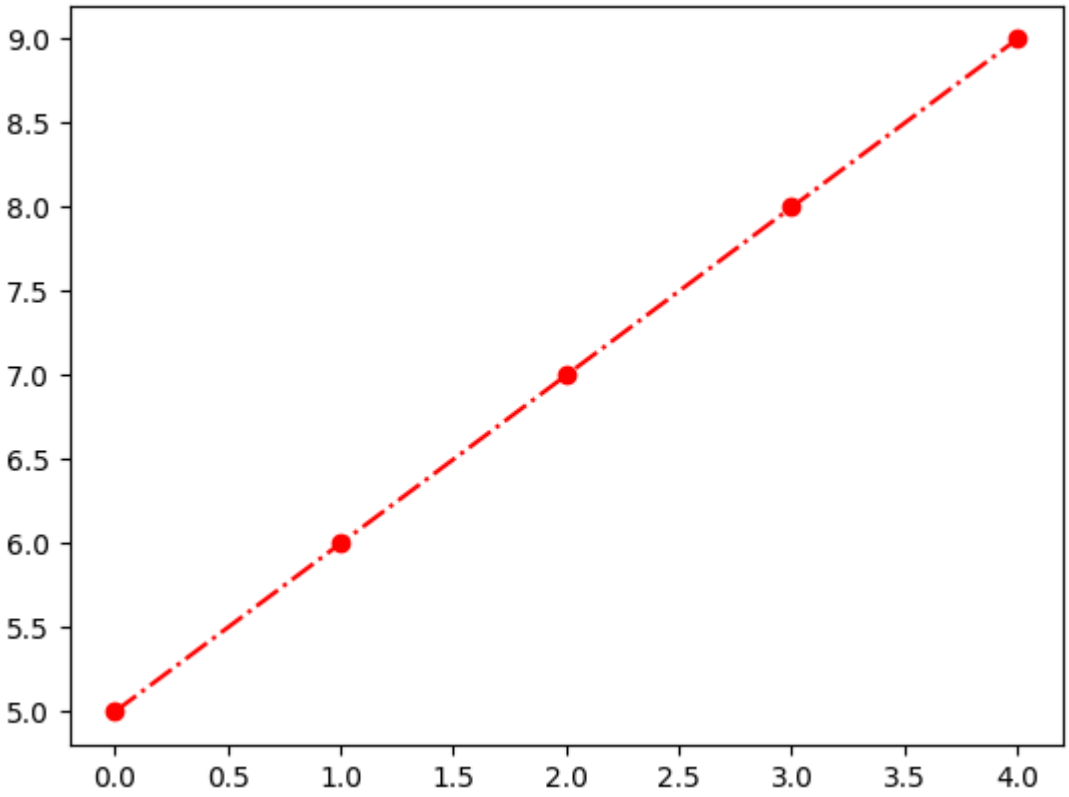
Add Paramaters to Plot

In [4]:
```
# Define the values for (x, y) to make graph
x = [0, 1, 2, 3, 4]
y = [5, 6, 7, 8, 9]
# Remember that length of both array must be equal

# Plot x vs y graph adding some parameters
plt.plot(x, y, color='red', linewidth='2', linestyle='-.', marker='.',
         markersize='10', markeredgecolor='blue')

plt.show() # Show graph
```



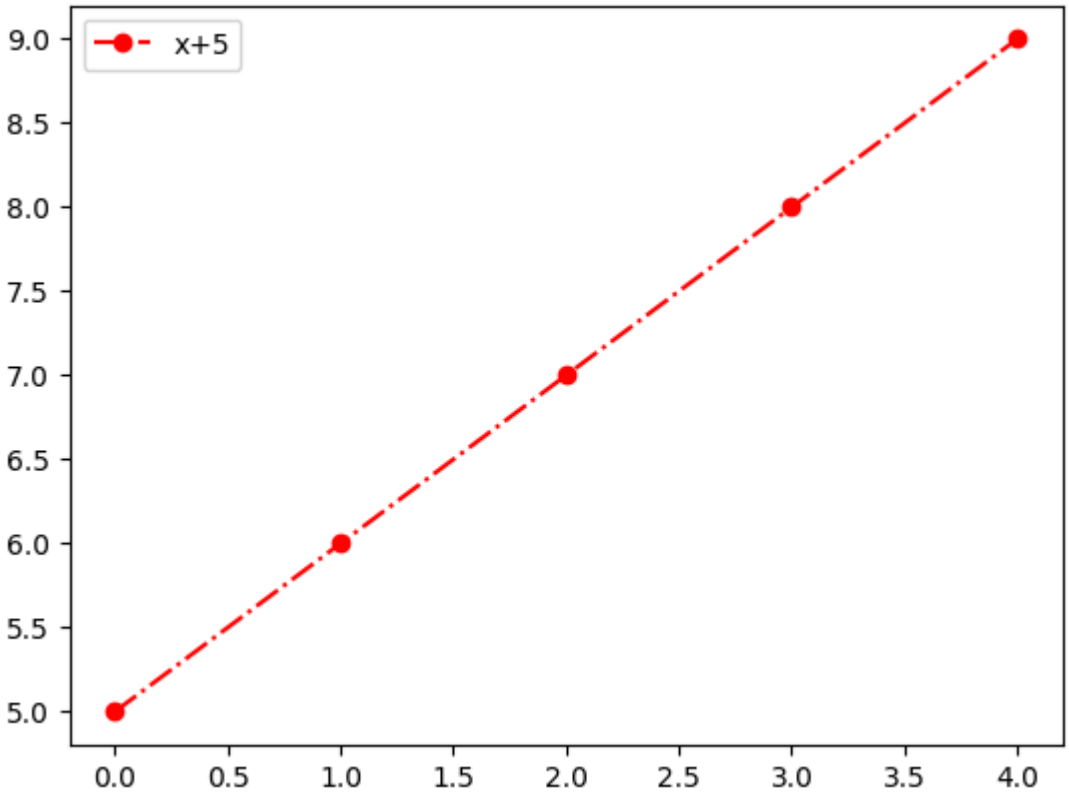
Use Shorthand Notation

In [5]:
```
# Define the values for (x, y) to make graph
x = [0, 1, 2, 3, 4]
y = [5, 6, 7, 8, 9]

# Shorthand notation format = '[color][marker][line]'
plt.plot(x, y, 'ro-.')

plt.show() # Show graph
```



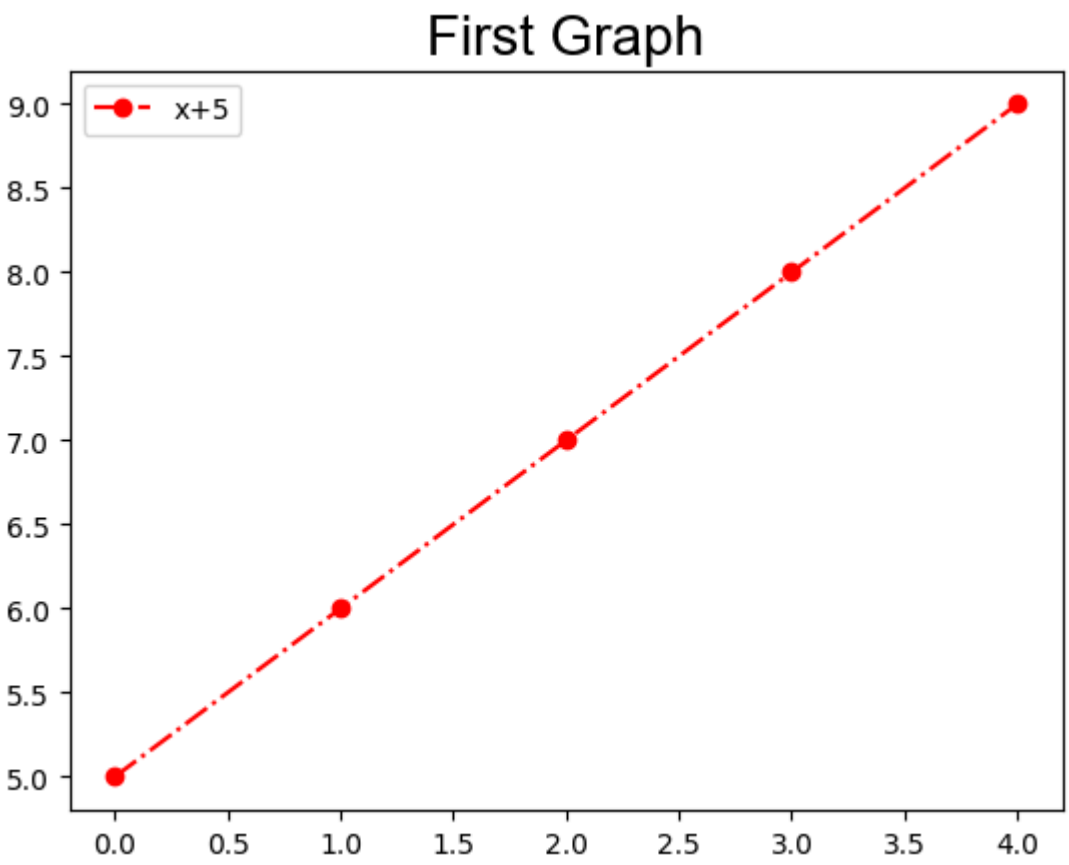
Add Legend

In [6]:
```
# Define the values for (x, y) to make graph
x = [0, 1, 2, 3, 4]
y = [5, 6, 7, 8, 9]

# Add label in plot function
plt.plot(x, y, 'ro-.', label='x+5')

plt.legend() # Show legend (label of graph)
plt.show() # Show graph
```



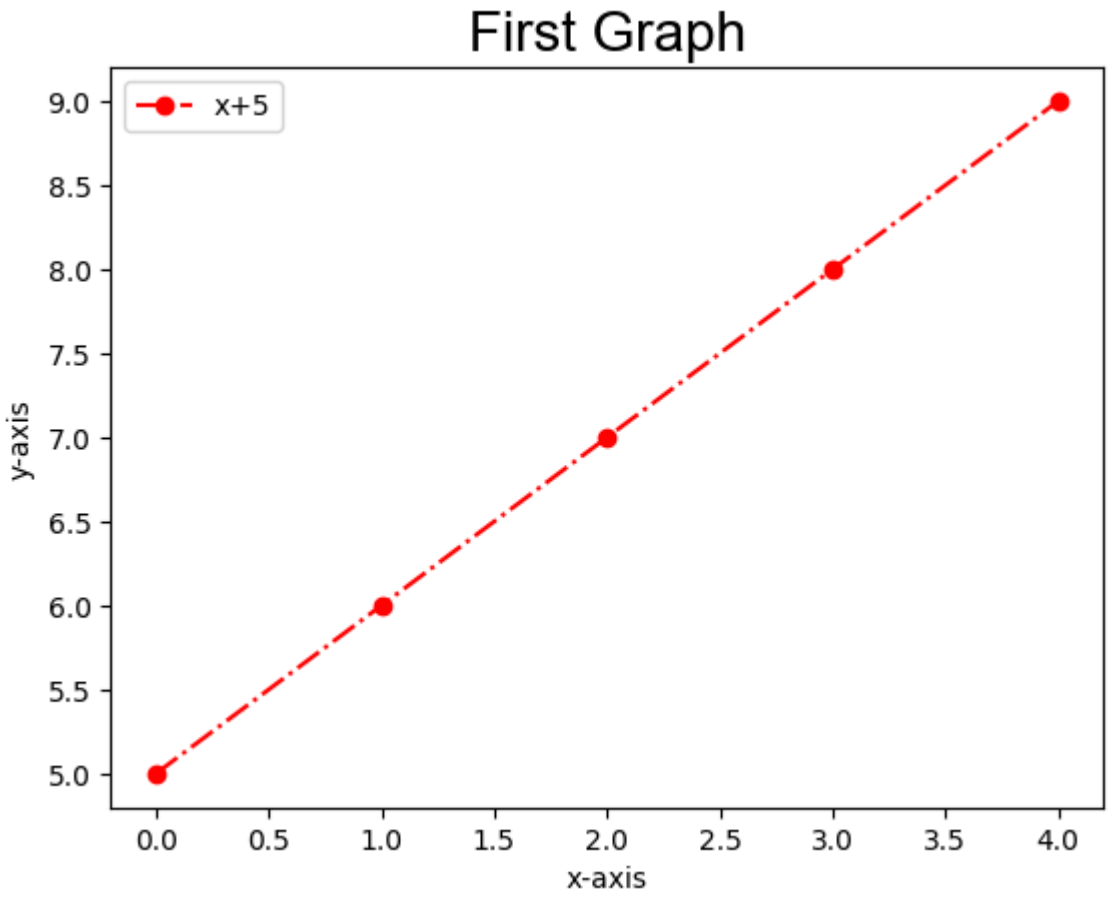
Add Title

In [7]:
```
# Define the values for (x, y) to make graph
x = [0, 1, 2, 3, 4]
y = [5, 6, 7, 8, 9]
# Remember that length of both array must be equal

# Plot x vs y graph
plt.plot(x, y, 'ro-.', label='x+5')

# Give title
plt.title('First Graph', fontdict={'fontname': 'Arial', 'fontsize': 20})
# fontdict is not mandatory

plt.legend() # Show legend (label of graph)
plt.show() # Show graph
```



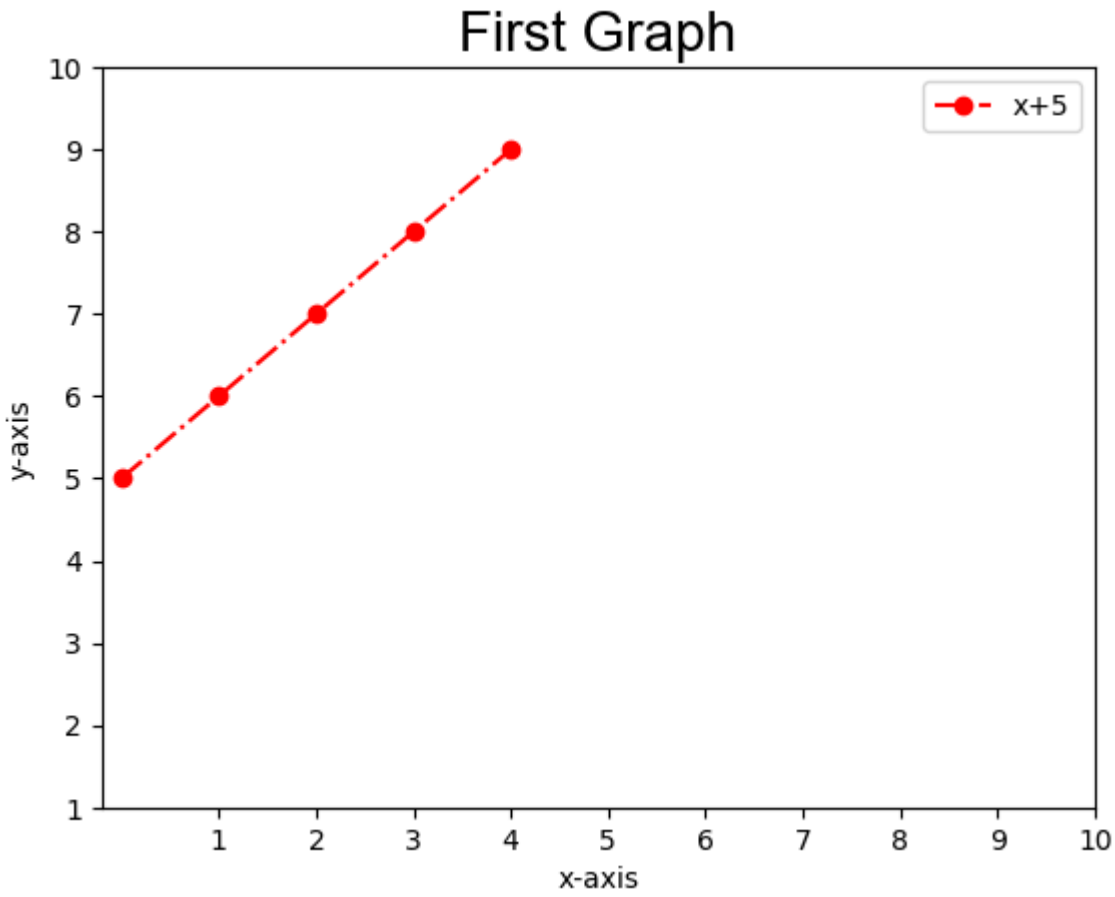
Add Axis Labels

In [8]:
```
# Define the values for (x, y) to make graph
x = [0, 1, 2, 3, 4]
y = [5, 6, 7, 8, 9]
# Remember that length of both array must be equal

# Plot x vs y graph
plt.plot(x, y, 'ro-.', label='x+5')

# Give title
plt.title('First Graph', fontdict={'fontname': 'Arial', 'fontsize': 20})
# fontdict is not mandatory

# Give labels to x-axis and y-axis
plt.xlabel('x-axis') # Label x-axis
plt.ylabel('y-axis') # Label y-axis
# fontdict can also be used

plt.legend() # Show legend
plt.show() # Show graph
```



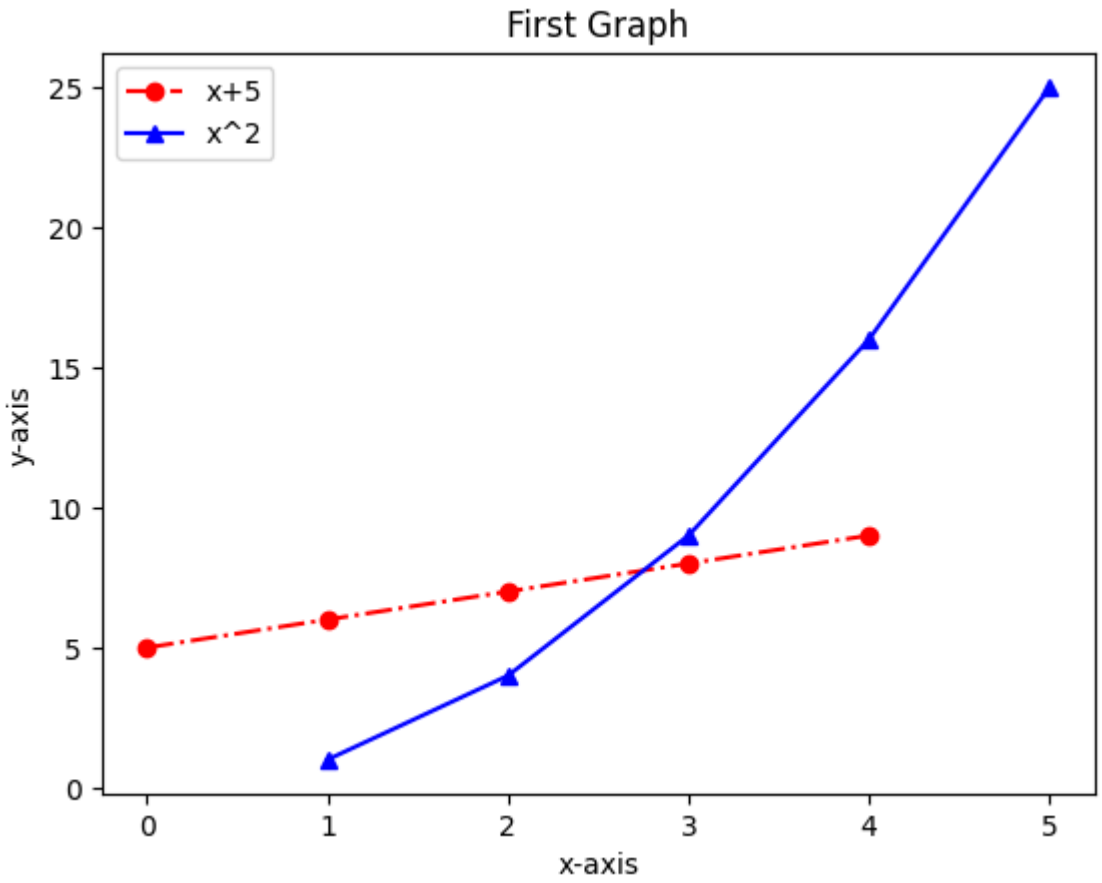
Change Ticks

In [9]:
```
# Define the values for (x, y) to make graph
x = [0, 1, 2, 3, 4]
y = [5, 6, 7, 8, 9]
# Remember that length of both array must be equal

# Plot x vs y graph
plt.plot(x, y, 'ro-.', label='x+5')

# Give title
plt.title('First Graph', fontdict={'fontname': 'Arial', 'fontsize': 20})
# fontdict is not mandatory

# Give labels to x-axis and y-axis
plt.xlabel('x-axis') # Label x-axis
plt.ylabel('y-axis') # Label y-axis
# fontdict can also be used

# Change the ticks
plt.xticks([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
plt.yticks([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
# Any values in-between can be also added if needed

plt.legend() # Show legend
plt.show() # Show graph
```



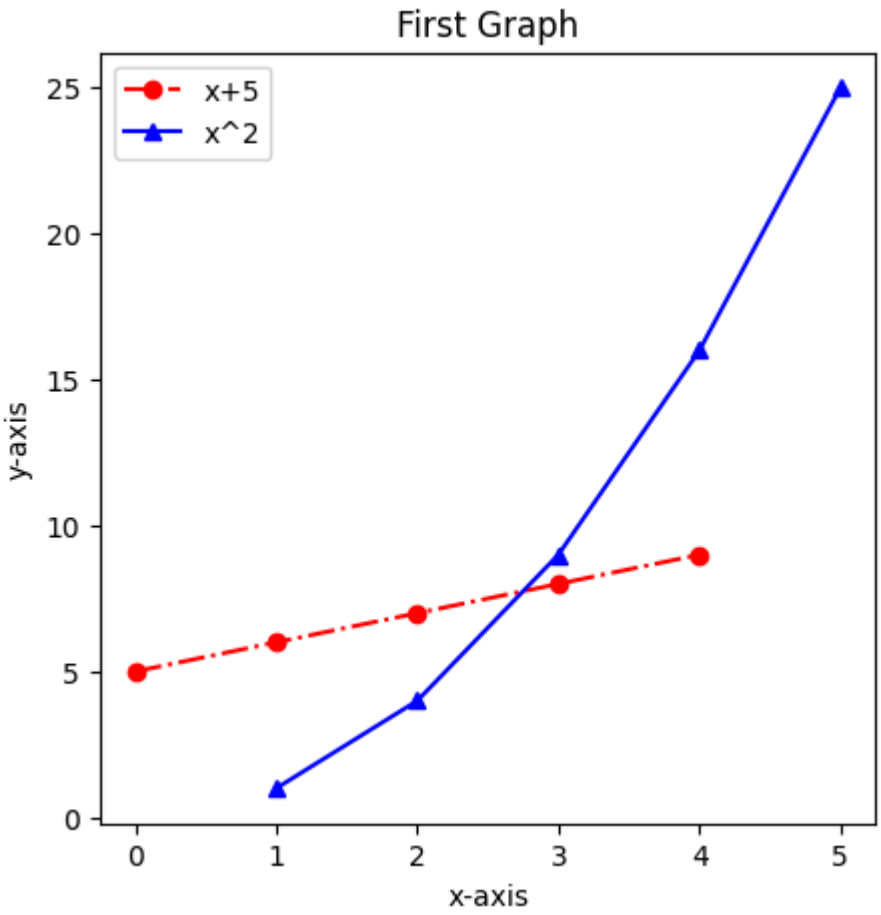
Add Another Graph

In [10]:
```
# Plot first graph
x = [0, 1, 2, 3, 4]
y = [5, 6, 7, 8, 9]
plt.plot(x, y, 'ro-.', label='x+5')

# Add another graph
x2 = [1, 2, 3, 4, 5]
y2 = [1, 4, 9, 16, 25]
plt.plot(x2, y2, 'b^-', label='x^2')

# Give title
plt.title('First Graph')

# Give labels to axes
plt.xlabel('x-axis')
plt.ylabel('y-axis')

plt.legend() # Show legend
plt.show() # Show graph
```



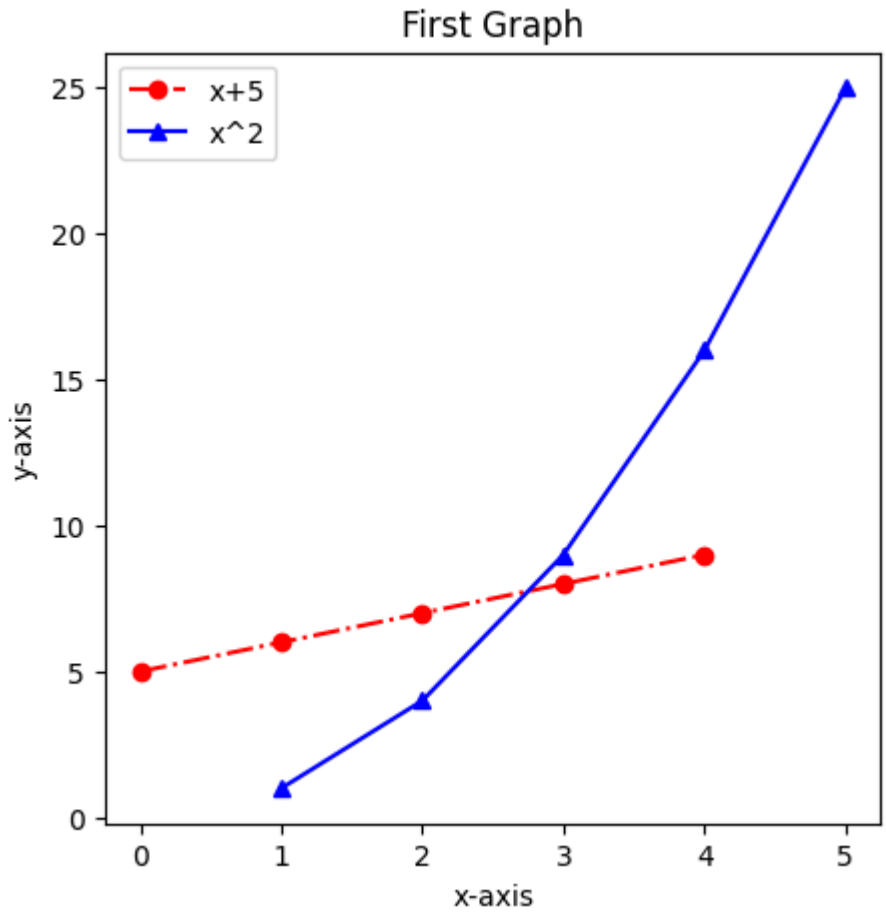
Resize Graph

In [11]:
```
# Resize graph
plt.figure(figsize=(5, 5), dpi=100)
# dpi is pixles per inch

# Plot graph
x = [0, 1, 2, 3, 4]
y = [5, 6, 7, 8, 9]
plt.plot(x, y, 'ro-.', label='x+5')
x2 = [1, 2, 3, 4, 5]
y2 = [1, 4, 9, 16, 25]
plt.plot(x2, y2, 'b^-', label='x^2')

# Give title
plt.title('First Graph')

# Give labels to axes
plt.xlabel('x-axis')
plt.ylabel('y-axis')

plt.legend() # Show legend
plt.show() # Show graph
```



Save Graph

In [12]:
```
# Resize graph
plt.figure(figsize=(5, 5), dpi=100)
# dpi is pixles per inch

# Plot graph
x = [0, 1, 2, 3, 4]
y = [5, 6, 7, 8, 9]
plt.plot(x, y, 'ro-.', label='x+5')
x2 = [1, 2, 3, 4, 5]
y2 = [1, 4, 9, 16, 25]
plt.plot(x2, y2, 'b^-', label='x^2')

# Give title
plt.title('First Graph')

# Give labels to axes
plt.xlabel('x-axis')
plt.ylabel('y-axis')

# Save graph
plt.savefig('my-graph.png', dpi=500)
# dpi is not mandatory

plt.legend()
plt.show()
```



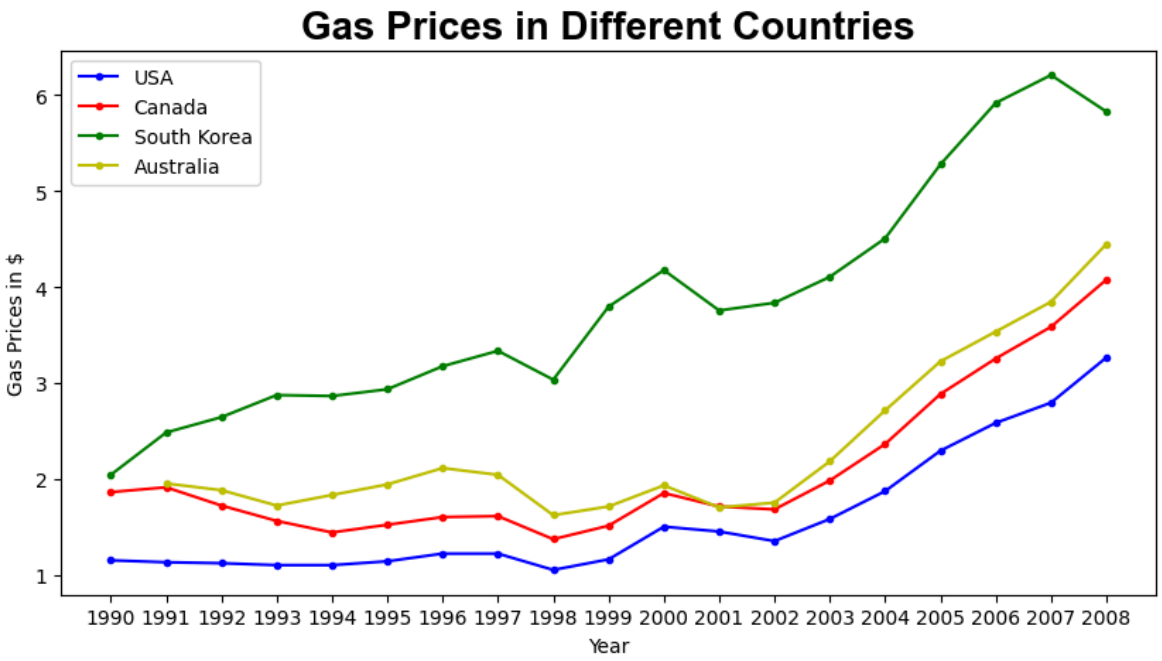
Real World Example

In [13]:
```
gas_prices = pd.read_csv('./assets/gas-prices.csv') # Read csv file
print(gas_prices.head()) # Show first 5 rows of data
```

```
   Year  Australia  Canada  France  Germany  Italy  Japan  Mexico  \
0  1990        NaN    1.87    3.63     2.65   4.59   3.16    1.00
1  1991       1.96    1.92    3.45     2.90   4.50   3.46    1.30
2  1992       1.89    1.73    3.56     3.27   4.53   3.58    1.50
3  1993       1.73    1.57    3.41     3.07   3.68   4.16    1.56
4  1994       1.84    1.45    3.59     3.52   3.70   4.36    1.48

   South Korea    UK   USA
0         2.05  2.82  1.16
1         2.49  3.01  1.14
2         2.65  3.06  1.13
3         2.88  2.84  1.11
4         2.87  2.99  1.11
```

In [14]:
```
# Define the figure size
plt.figure(figsize=(10, 5), dpi=100)
# 10 inches wide and 5 inches tall

# Give the title
plt.title(
    'Gas Prices in Different Countries',
    fontdict={
        'fontsize': 20,
        'fontweight': 'bold',
        'fontname': 'Arial'
    }
```

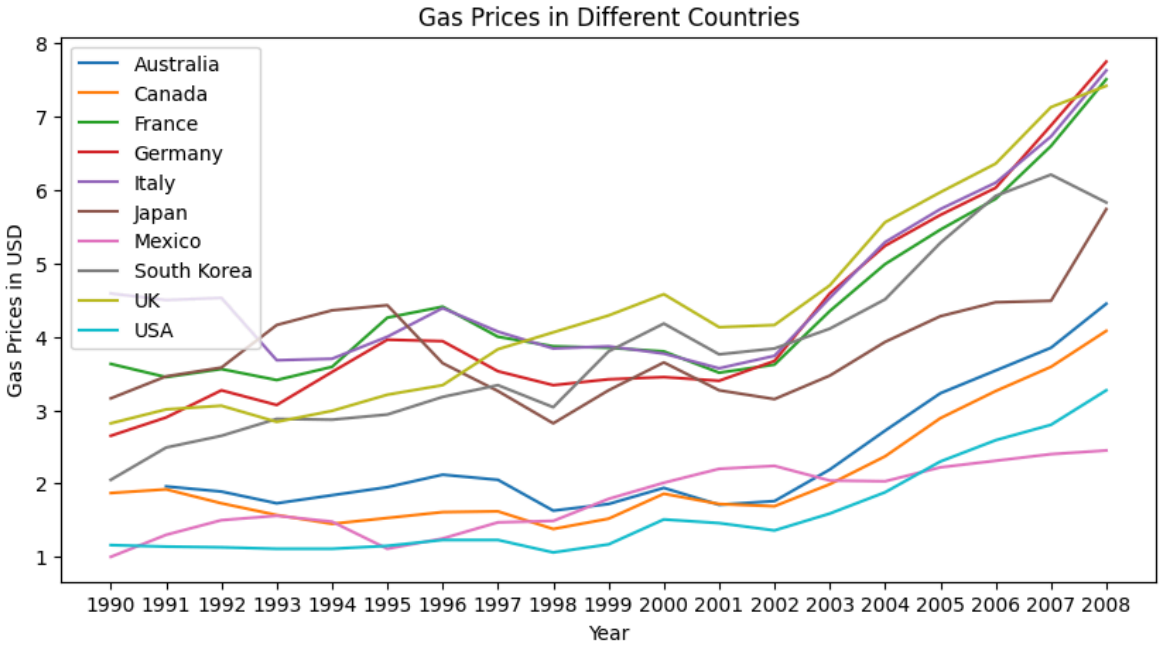
```
)

# Plot the line graphs
plt.plot(gas_prices.Year, gas_prices.USA, 'b.-', label='USA')
plt.plot(gas_prices.Year, gas_prices.Canada, 'r.-', label='Canada')
plt.plot(gas_prices.Year, gas_prices['South Korea'], 'g.-', label='South Korea')
# [] is used to access column with space in name
# You can also use dot notation to access column name without space
# But for the column with space, you have to use [] notation
plt.plot(gas_prices.Year, gas_prices['Australia'], 'y.-', label='Australia')
# You can also use a list of selected countries to plot
# plt.plot(gas_prices.Year, gas_prices[['USA', 'Canada', 'South Korea', 'Austral

# Set the x-axis ticks for all the years having data
plt.xticks(gas_prices.Year)
# If we want the x-axis ticks to be in 5 years interval
# plt.xticks(gas_prices.Year[::5])

# Set the x labels and y labels
plt.xlabel('Year')
plt.ylabel('Gas Prices in $')

# Save the graph
plt.savefig('gas-prices.png', dpi=500)

plt.legend() # Show legend (label of graph)
plt.show()
```



If we want to plot line graph for all countries at once.

In [15]:
```
# Define the figure size
plt.figure(figsize=(10, 5), dpi=100) # 10 inches wide and 5 inches tall

# Set the title
plt.title('Gas Prices in Different Countries')

# Set the x-axis ticks for all the years having data
plt.xticks(gas_prices.Year)
```

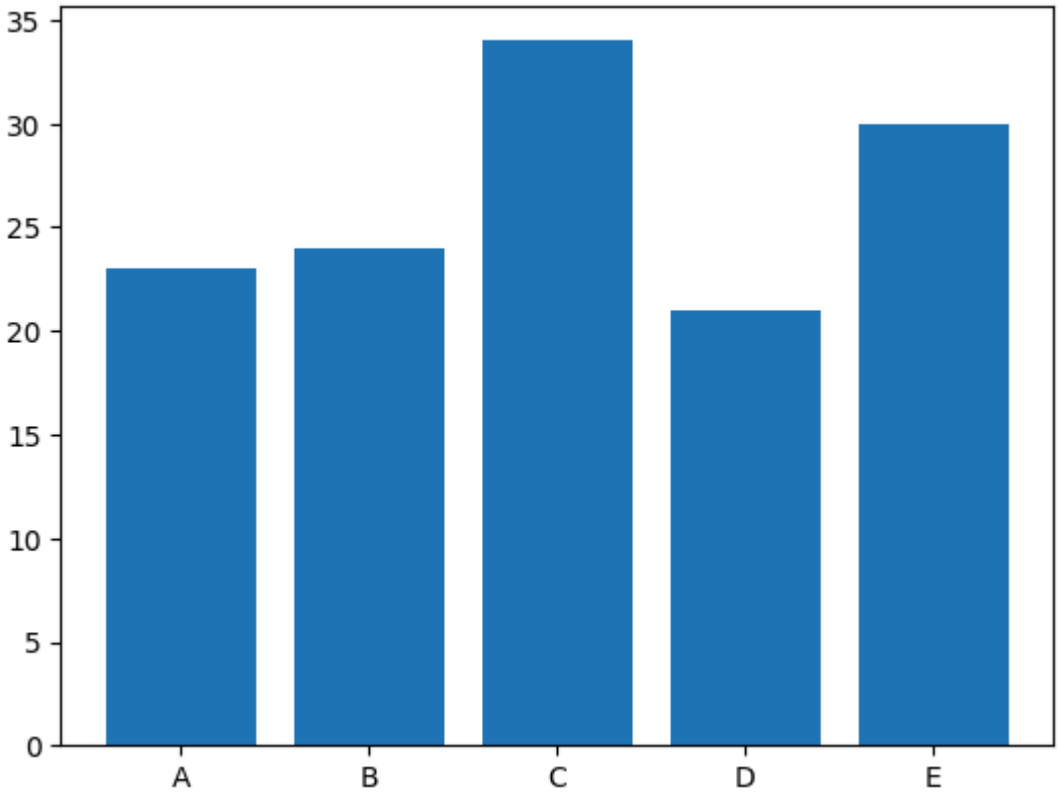
```
# Set the xlabel and ylabel
plt.xlabel('Year')
plt.ylabel('Gas Prices in USD')

# Plot the line graphs
for country in gas_prices.columns[1:]:
    plt.plot(gas_prices.Year, gas_prices[country], label=country)


plt.legend() # Show legend (label of graph)
plt.show() # Show graph
```



Bar Charts

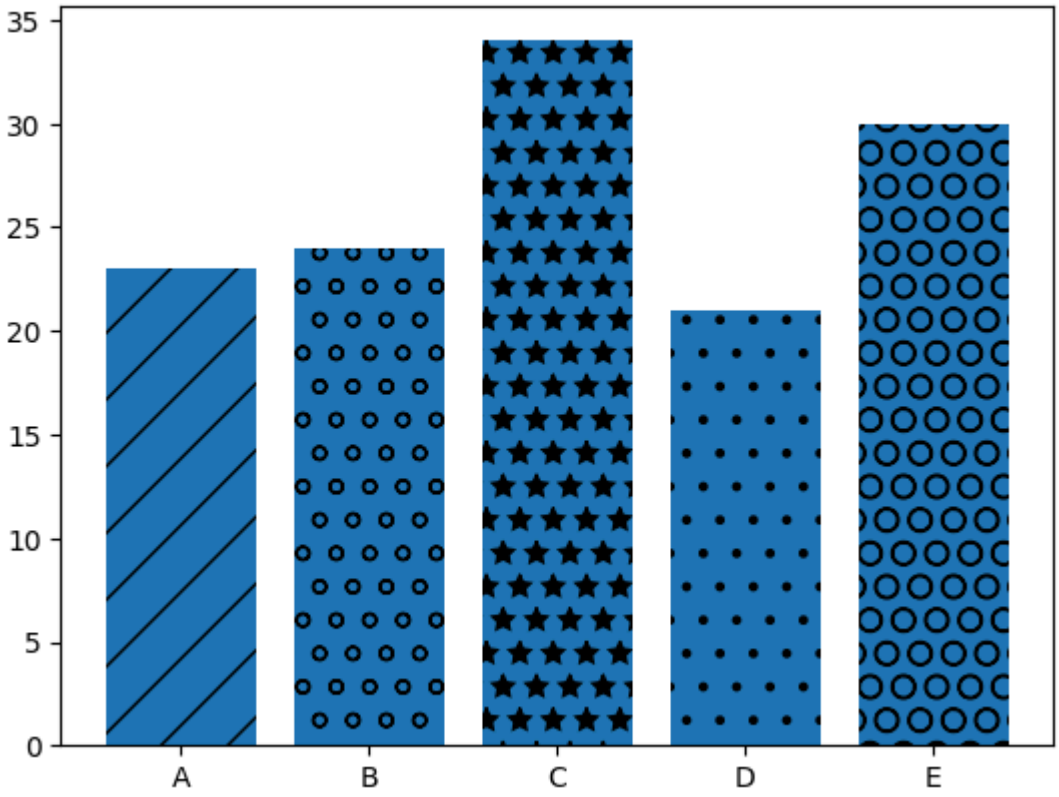
Basic Plot

In [16]:
```
# Define labels and values
labels = ['A', 'B', 'C', 'D', 'E']
values = [23, 24, 34, 21, 30]
# Remember each labels should have a value i.e length of labels and values must

# Create a bar graph
plt.bar(labels, values)

plt.show() # Show graph
```



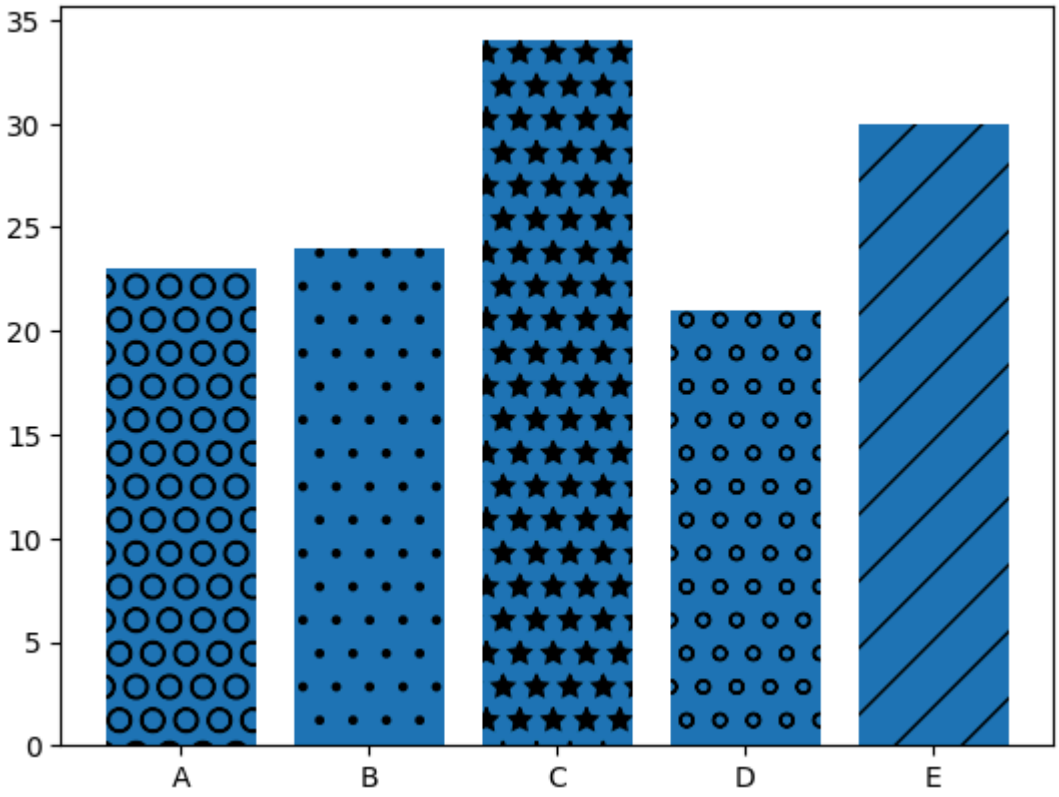
Customize Bars

In [17]:
```
# Define labels and values
labels = ['A', 'B', 'C', 'D', 'E']
values = [23, 24, 34, 21, 30]
# Remember each labels should have a value
# i.e length of labels and values must be equal

# Create a bar graph
bars = plt.bar(labels, values)

# Customize bars
bars[0].set_hatch('/')
bars[1].set_hatch('o')
bars[2].set_hatch('*')
bars[3].set_hatch('.')
bars[4].set_hatch('O')

plt.show() # Show graph
```



Customize Bars Using Loop

In [18]:
```
# Define labels and values
labels = ['A', 'B', 'C', 'D', 'E']
values = [23, 24, 34, 21, 30]
# Remember each labels should have a value
# i.e length of labels and values must be equal

# Create a bar graph
bars = plt.bar(labels, values)

# Customize bars
patterns = ['/', 'o', '*', '.', 'O'] # Bar styles
for bar in bars:
    bar.set_hatch(patterns.pop())

plt.show() # Show graph
```

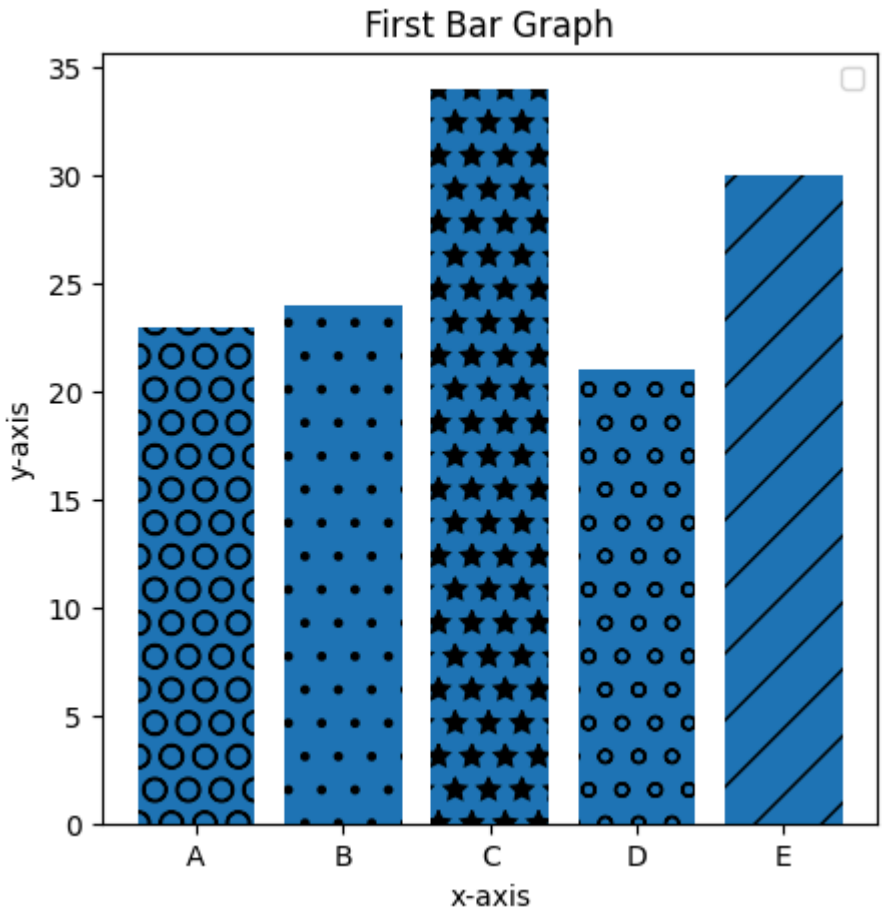


Adding Features

We can add features in bar graph as in line graph

In [19]:
```
# Resize graph
plt.figure(figsize=(5, 5), dpi=100)
# dpi is pixles per inch

# Create bar graph
labels = ['A', 'B', 'C', 'D', 'E']
values = [23, 24, 34, 21, 30]
bars = plt.bar(labels, values)

# Customize bars
patterns = ['/', 'o', '*', '.', 'O'] # Bar styles
for bar in bars:
    bar.set_hatch(patterns.pop())

# Give title
plt.title('First Bar Graph')

# Give labels to axes
plt.xlabel('x-axis')
plt.ylabel('y-axis')

# Save graph
plt.savefig('my-bar-graph.png', dpi=500)
# dpi is not mandatory

plt.legend()
plt.show()
```

```
C:\Users\kusha\AppData\Local\Temp\ipykernel_3760\1594682148.py:26: UserWarning: N
o artists with labels found to put in legend.  Note that artists whose label star
t with an underscore are ignored when legend() is called with no argument.
  plt.legend()
```



Histograms

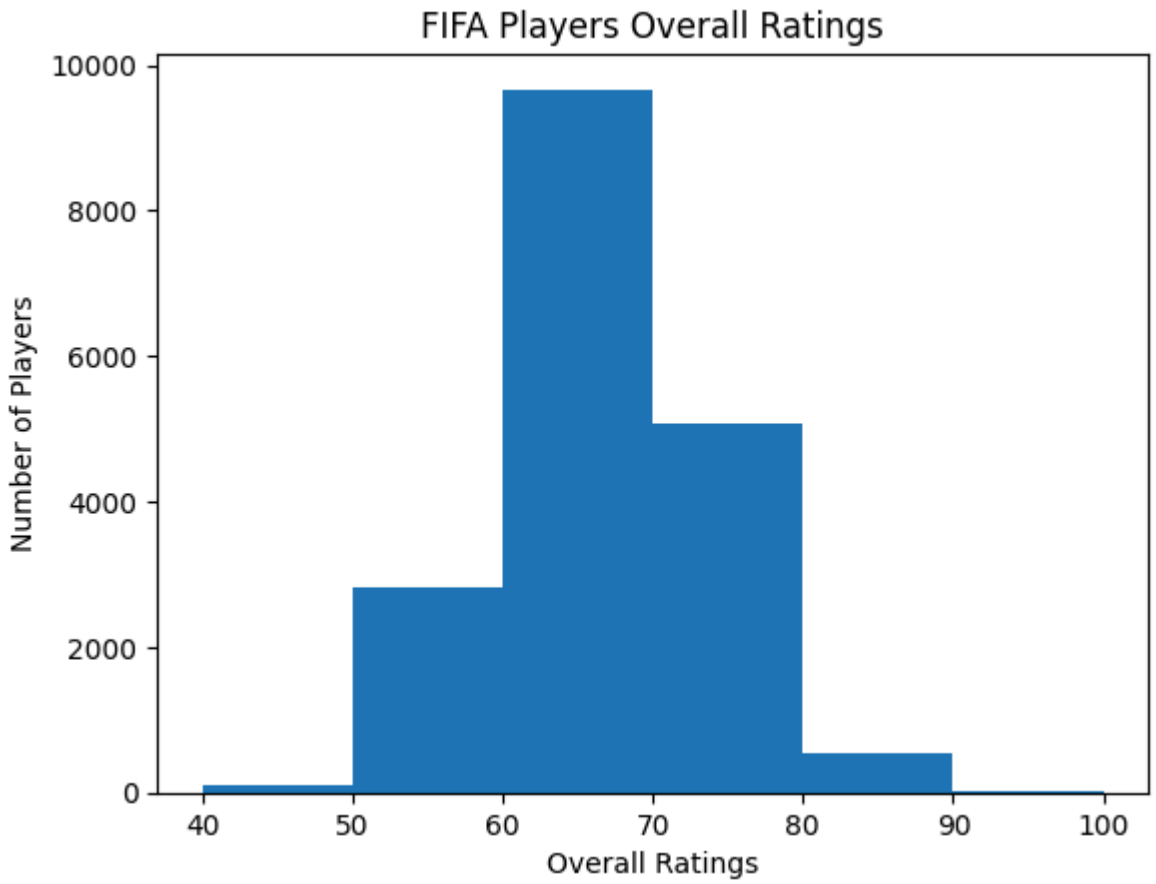
In [20]:
```
fifa_data = pd.read_csv('./assets/fifa-data.csv') # Read csv file
fifa_data.head() # Show first 5 rows of data
```

Out[20]:

	Unnamed: 0	ID	Name	Age	Photo	Natio
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Arge
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Po
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Be

5 rows × 89 columns

In [21]:
```
# Define bins
bins = [40, 50, 60, 70, 80, 90, 100]

# Give title
plt.title('FIFA Players Overall Ratings')
# You can also give the hex color code for the title
# plt.title('FIFA Players Overall Ratings', color='#FF5733')

# Create histogram
plt.hist(fifa_data.Overall, bins=bins)
# You can also use the following code to create histogram
# plt.hist(fifa_data.Overall, bins=bins, color='blue', edgecolor='black', alpha=
# alpha is the transparency of the color
# You can also give color name as hex code
# plt.hist(fifa_data.Overall, bins=bins, color='#FF5733', edgecolor='black', alp

# Set xlabel and ylabel
plt.xlabel('Overall Ratings')
plt.ylabel('Number of Players')

plt.show()
```



Pie Charts

In [22]:
```
fifa = pd.read_csv('./assets/fifa-data.csv') # Read csv file
fifa.head() # Show first 5 rows of data
```

Out[22]:

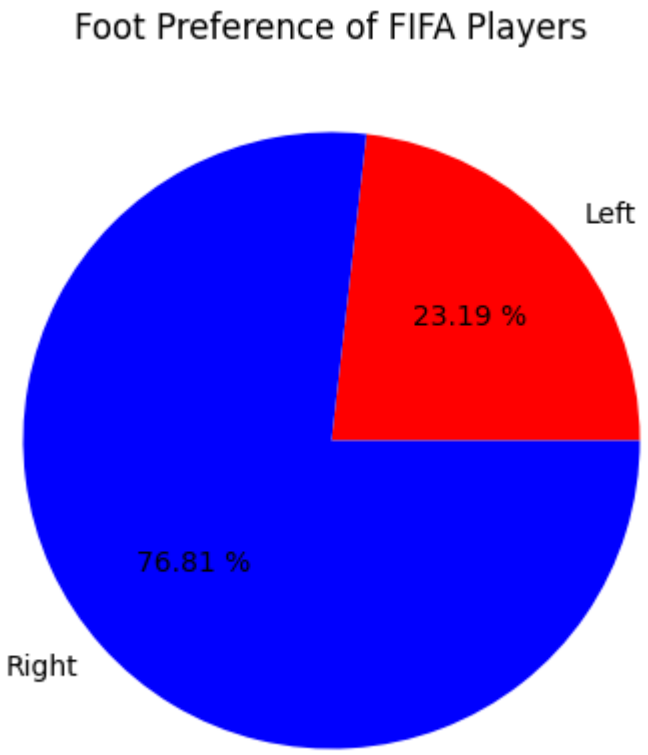
	Unnamed: 0	ID	Name	Age	Photo	Natio
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Arge
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Po
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Be

5 rows × 89 columns

Pie Chart for Preferred Foot by Player

In [23]:

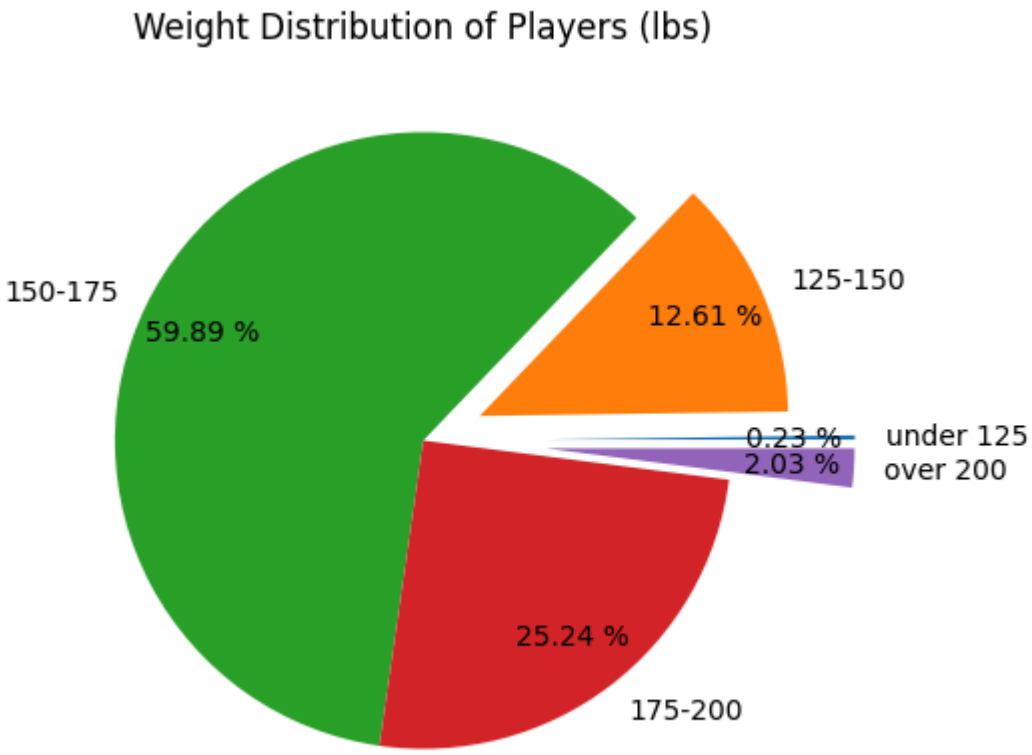
```
# Set the figure size
plt.figure(figsize=(8,5))

# Give title to pie chart
plt.title('Foot Preference of FIFA Players')

# Get the values for left and right foot preference
left = fifa[fifa['Preferred Foot'] == 'Left'].shape[0]
right = fifa[fifa['Preferred Foot'] == 'Right'].shape[0]

# Set labels and colors
labels = ['Left', 'Right']
colors = ['red', 'blue'] # You can also use hex codes for colors

# Create a pie chart
plt.pie([left, right], labels = labels, colors=colors, autopct='%.2f %%')

plt.show()
```



Pie Chart on Weight Distribution of Players

In [24]:

```
# Set the figure size
plt.figure(figsize=(8,5), dpi=100)

# Give title to pie chart
plt.title('Weight Distribution of Players (lbs)')

# Remove 'lbs' from the weight column and convert to int
fifa.Weight = [int(x.strip('lbs')) if type(x)==str else x for x in fifa.Weight]

# Get the values for different weight categories
# Count the number of players in each weight category
light = fifa.loc[fifa.Weight < 125].shape[0]
light_medium = fifa[(fifa.Weight >= 125) & (fifa.Weight < 150)].shape[0]
medium = fifa[(fifa.Weight >= 150) & (fifa.Weight < 175)].shape[0]
medium_heavy = fifa[(fifa.Weight >= 175) & (fifa.Weight < 200)].shape[0]
heavy = fifa[fifa.Weight >= 200].shape[0]

# Set the values for pie chart
weights = [light,light_medium, medium, medium_heavy, heavy]
label = ['under 125', '125-150', '150-175', '175-200', 'over 200']
explode = (.4,.2,0,0,.4)

# Create a pie chart
plt.pie(weights, labels=label, explode=explode, pctdistance=0.8, autopct='%.2f %

plt.show()
```



Box and Whiskers Chart

In [25]:

```
# Set the figure size
plt.figure(figsize=(5,8), dpi=100)

# Set the style of the plot
plt.style.use('default')

# Give title to the plot
plt.title('Team Comparison')

# Set ylabel
plt.ylabel('FIFA Overall Rating')

# Get the values for different teams
# Get the overall ratings for different teams
barcelona = fifa.loc[fifa.Club == "FC Barcelona"]['Overall']
madrid = fifa.loc[fifa.Club == "Real Madrid"]['Overall']
revs = fifa.loc[fifa.Club == "New England Revolution"]['Overall']

# Create a box plot
plt.boxplot(
    [barcelona, madrid, revs],
    labels=['FC Barcelona','Real Madrid','NE Revolution'],
    patch_artist=True,
    medianprops={'linewidth': 2}
)

plt.show()
```

```
C:\Users\kusha\AppData\Local\Temp\ipykernel_3760\3643175673.py:20: MatplotlibDepr
ecationWarning: The 'labels' parameter of boxplot() has been renamed 'tick_label
s' since Matplotlib 3.9; support for the old name will be dropped in 3.11.
  plt.boxplot(
```