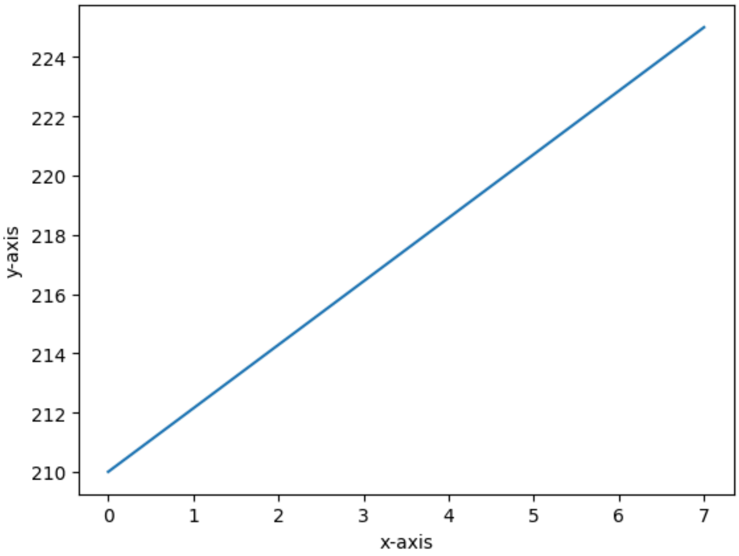
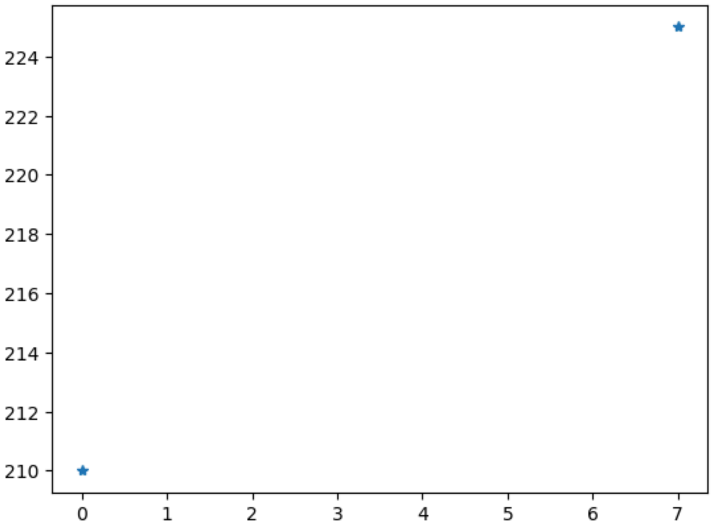
```
#draw a line in a diagram from position (0,0) to position (6,250):
import matplotlib.pyplot as plt
import numpy as np
x = np.array([0,7])
y = np.array([210,225])
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.plot(x,y)
plt.show()
```

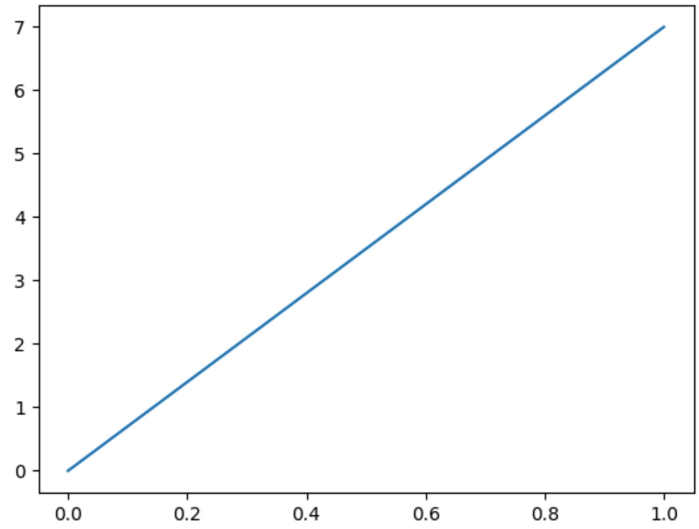


```
plt.plot(x,y,'*')
plt.show()
```



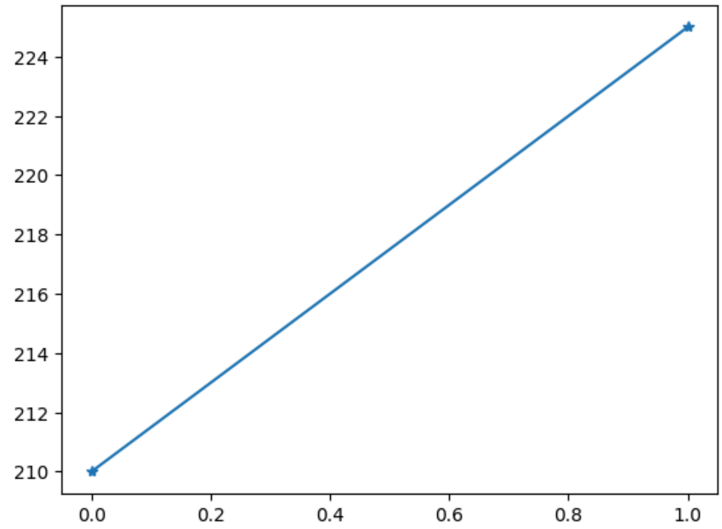
```
plt.plot(x)
```

```
[<matplotlib.lines.Line2D at 0x7d54151bdcf0>]
```



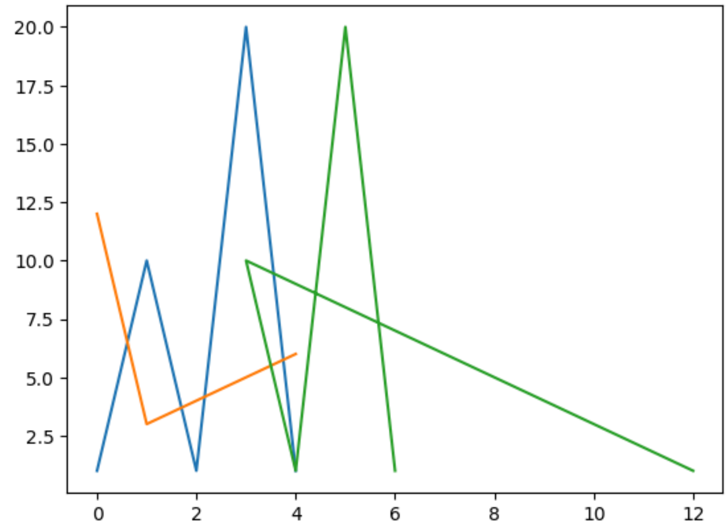
```
plt.plot(y,marker='*')
```

```
[<matplotlib.lines.Line2D at 0x7d54146be6b0>]
```

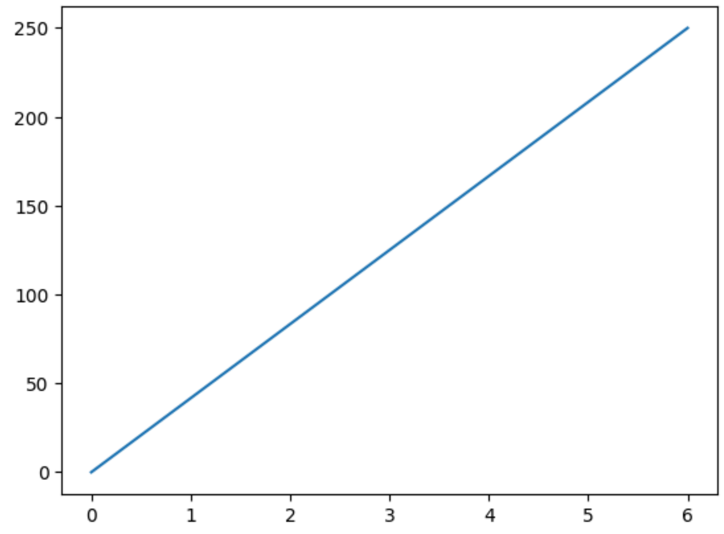


```
x = np.array([12,3,4,5,6])
y = np.array([1,10,1,20,1])
plt.plot(y)
plt.plot(x)
plt.plot(x,y)
```

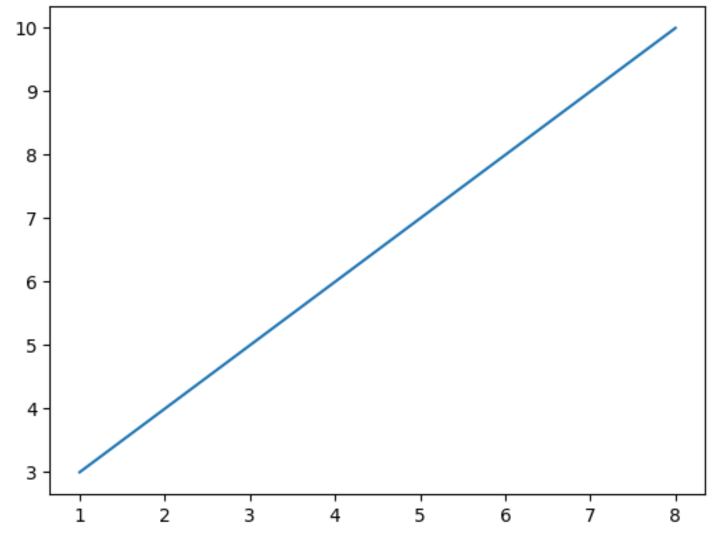
```
[<matplotlib.lines.Line2D at 0x7d54147d3400>]
```



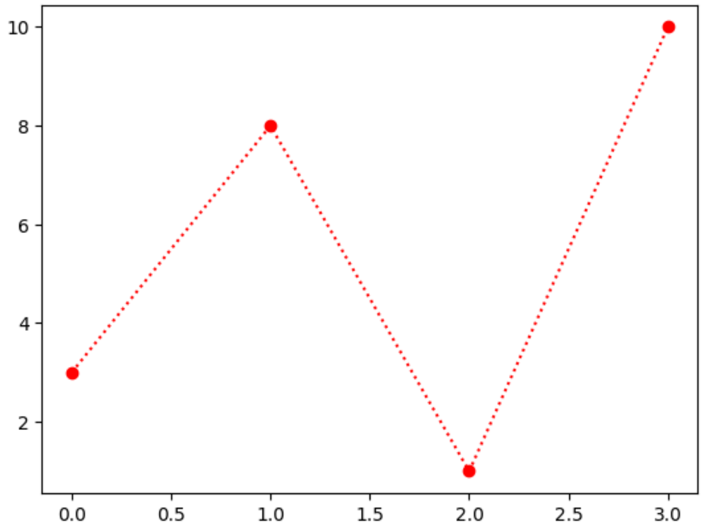
```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```



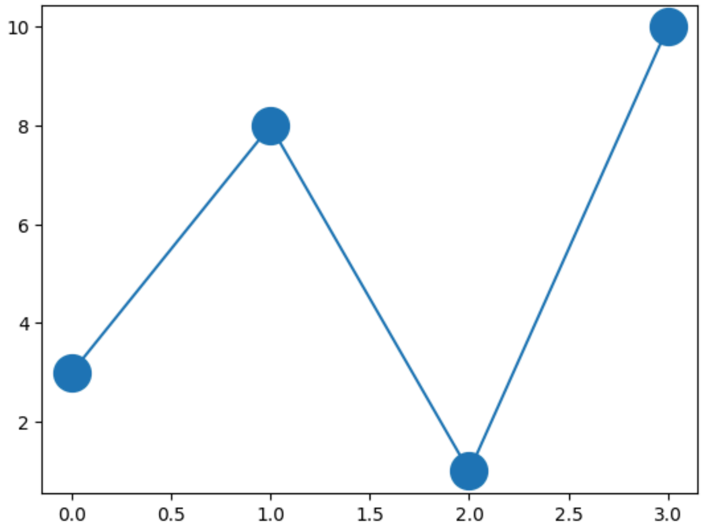
```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints)
plt.show()
```



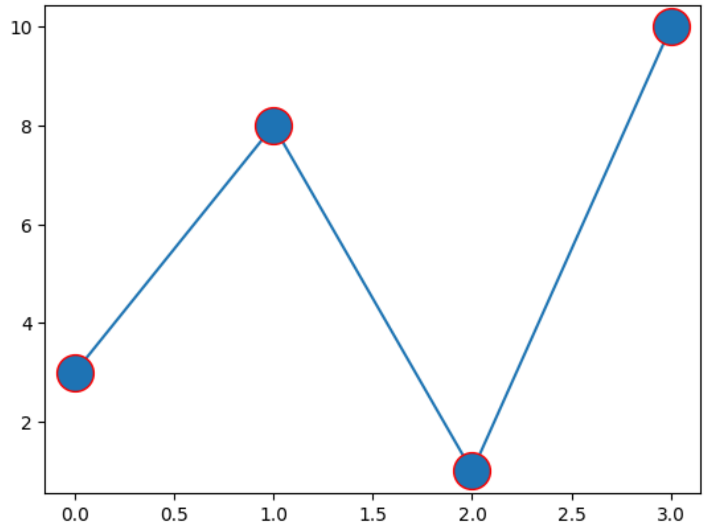
```
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, 'o:r')
plt.show()
```



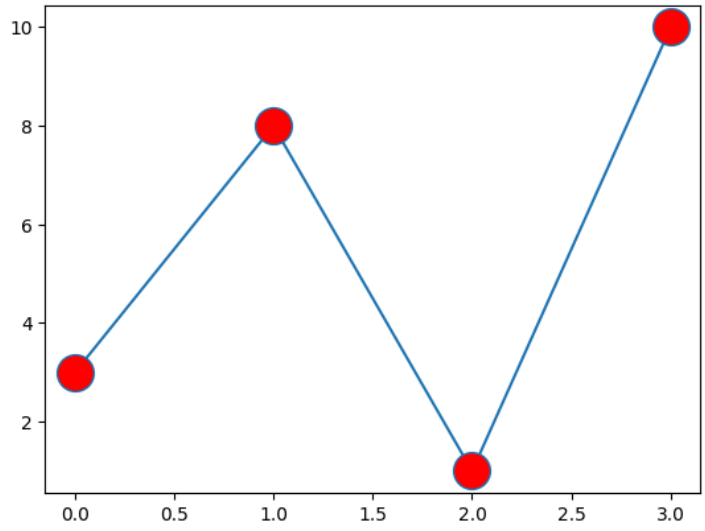
```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20)
plt.show()
```



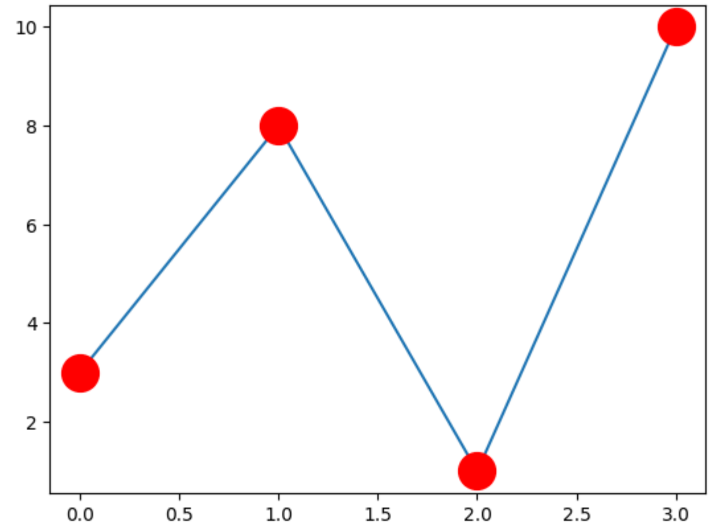
```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mec = 'r')
plt.show()
```



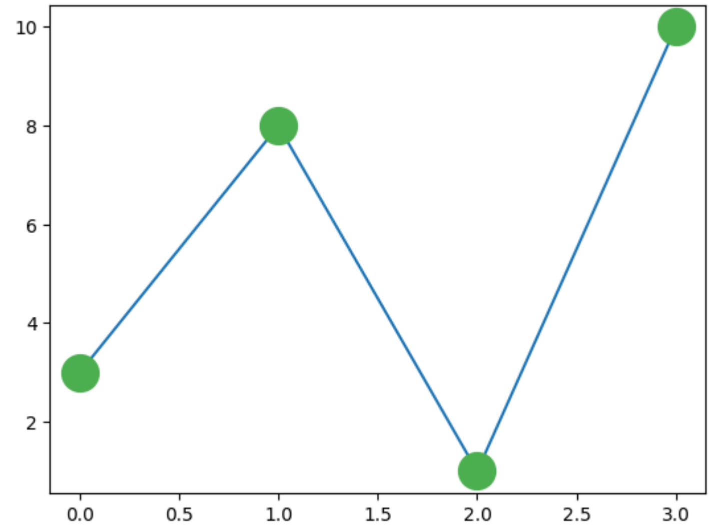
```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mfc = 'r')
plt.show()
```



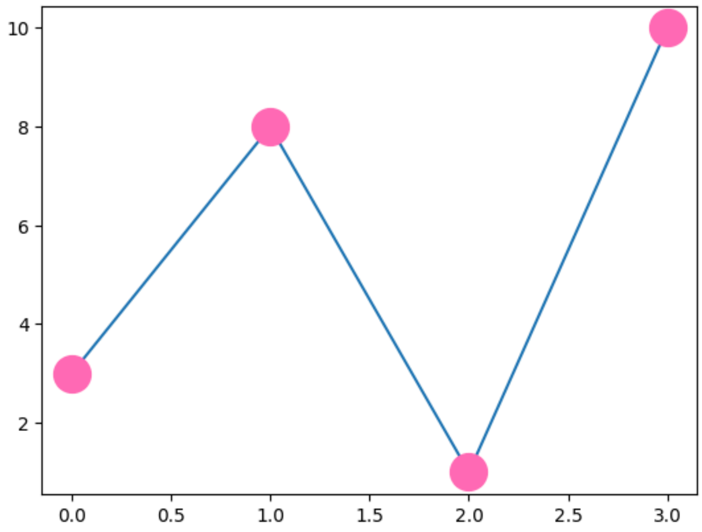
```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mec = 'r', mfc = 'r')
plt.show()
```



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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mec = '#4CAF50', mfc = '#4CAF50')
plt.show()
```



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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 20, mec = 'hotpink', mfc = 'hotpink')
plt.show()
```

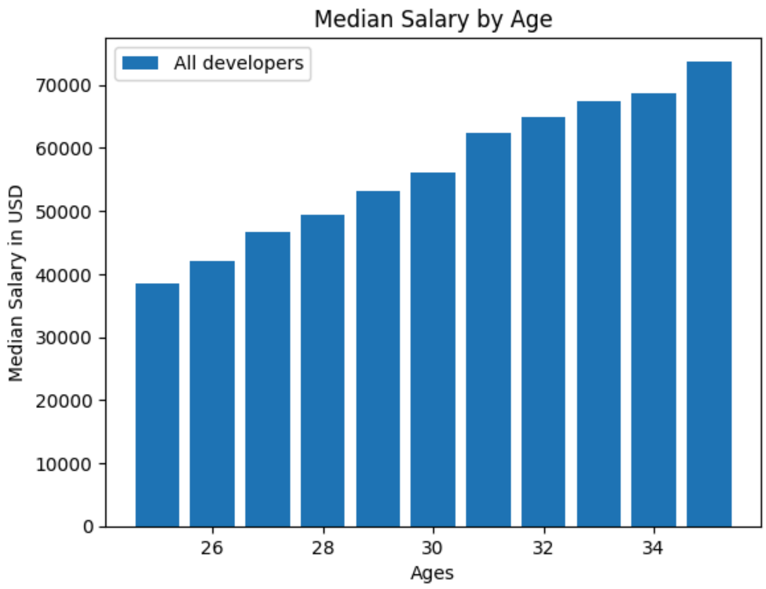


Bar_Charts

```
import pandas as pd
import matplotlib.pyplot as plt
x = [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]
devs_y = [38496, 42000, 46752, 49320, 53200, 56000, 62316, 64928, 67317, 68748, 73752]
```

Plotting the bar plot

```
plt.bar(x, devs_y, label="All developers")
plt.xlabel("Ages")
plt.ylabel("Median Salary in USD")
plt.title("Median Salary by Age")
plt.legend()
plt.show()
```

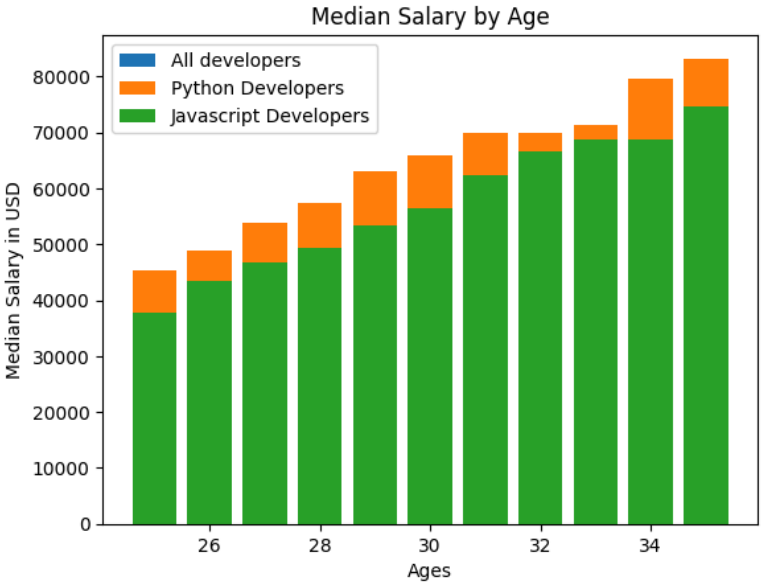


2. Adding more bars to the same plot

```
py_devs_y = [45372, 48876, 53850, 57287, 63016, 65998, 70003, 70000, 71418, 79674, 83238]
```

```
js_devs_y = [37810, 43515, 46823, 49293, 53437, 56373, 62375, 66674, 68745, 68746, 74583]
```

```
plt.bar(x, devs_y, label="All developers")
plt.bar(x, py_devs_y, label="Python Developers")
plt.bar(x, js_devs_y, label="Javascript Developers")
plt.xlabel("Ages")
plt.ylabel("Median Salary in USD")
plt.title("Median Salary by Age")
plt.legend()
plt.show()
```

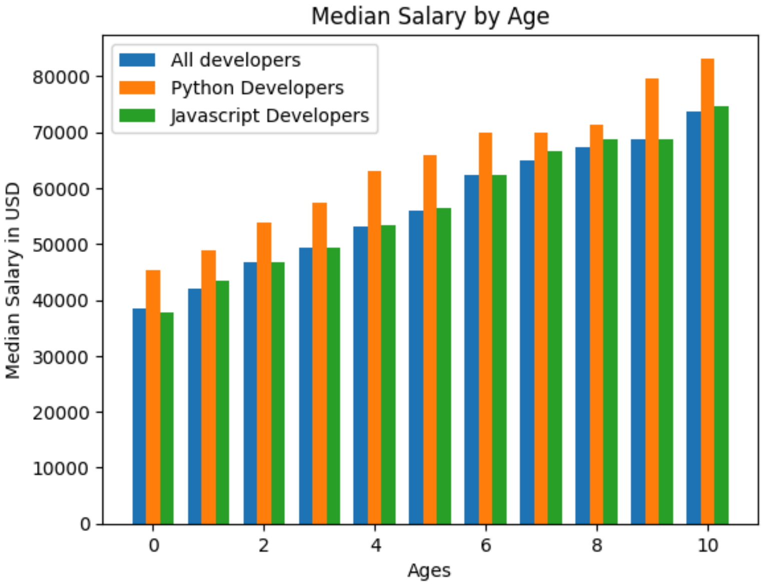


3. Adjusting the width of the plot

```
import numpy as np

x_indexes = np.arange(len(x))
```

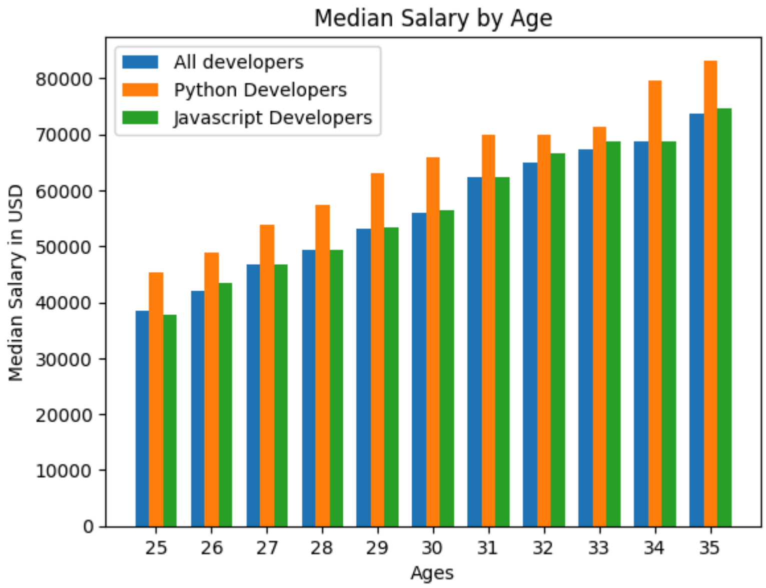
```
width = 0.25
```

```
plt.bar(x_indexes - width, devs_y, width=width, label="All developers")
plt.bar(x_indexes, py_devs_y, width = width, label="Python Developers")
plt.bar(x_indexes + width, js_devs_y, width=width, label="Javascript Developers")
plt.xlabel("Ages")
plt.ylabel("Median Salary in USD")
plt.title("Median Salary by Age")
plt.legend()
plt.show()
```



4. Changing the xlabels

```
plt.bar(x_indexes - width, devs_y, width=width, label="All developers")
plt.bar(x_indexes, py_devs_y, width = width, label="Python Developers")
plt.bar(x_indexes + width, js_devs_y, width=width, label="Javascript Developers")
plt.xlabel("Ages")
plt.ylabel("Median Salary in USD")
plt.title("Median Salary by Age")
plt.xticks(ticks=x_indexes, labels=x) #changing the xlabel
plt.legend()
plt.show()
```



5. Plotting the bar plot from pandas dataframe

```
import pandas as pd
data = pd.read_csv('/content/data.csv')
```

```
from collections import Counter
```

```
ids = data['Responder_id']
language_responses = data['LanguagesWorkedWith']
```

```
language_counter = Counter()
```

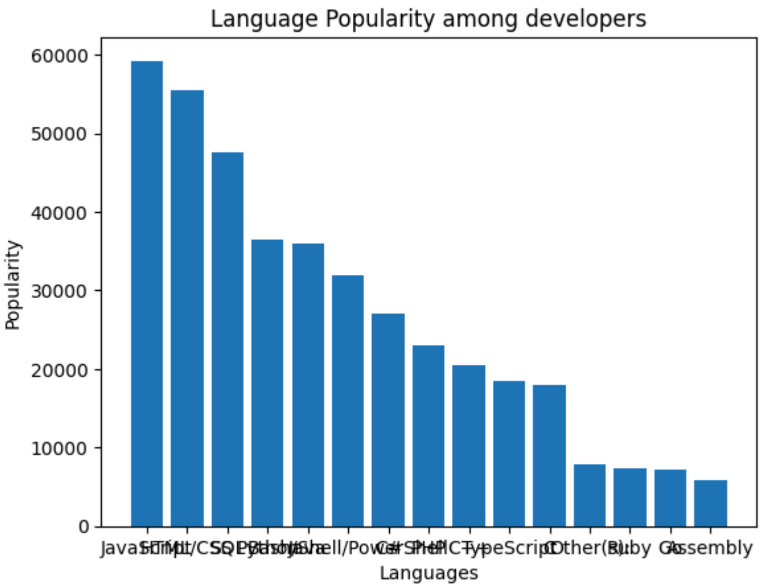
```
for response in language_responses:
    language_counter.update(response.split(";"))
```

```
languages = []
popularity = []

for item in language_counter.most_common(15):
    languages.append(item[0])
    popularity.append(item[1])
```

```
print(languages)
print(popularity)
```

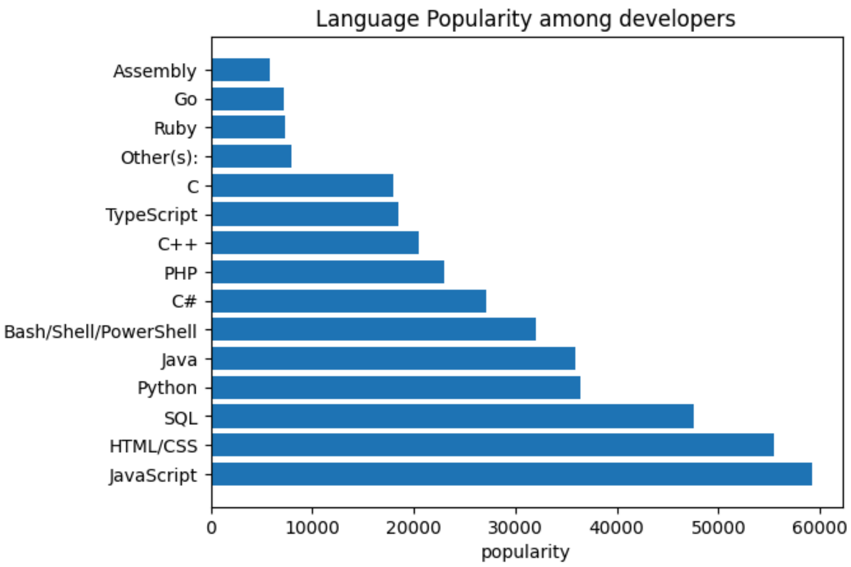
```
['JavaScript', 'HTML/CSS', 'SQL', 'Python', 'Java', 'Bash/Shell/PowerShell', 'C#', 'PHP', 'C++', 'TypeScript', 'C', 'Other(s):', 'Rub
[59219, 55466, 47544, 36443, 35917, 31991, 27097, 23030, 20524, 18523, 18017, 7920, 7331, 7201, 5833]
```

```
plt.bar(languages, popularity)
plt.xlabel("Languages")
plt.ylabel("Popularity")
plt.title("Language Popularity among developers")
plt.show()
```



6. Plotting Horizontal bar chart

```
plt.barh(languages, popularity)
plt.xlabel("popularity")
plt.title("Language Popularity among developers")
plt.show()
```



```
languages.reverse()
popularity.reverse()
```

https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages

plt.figure(figsize=(8,6)) plt.barh(languages, popularity) plt.xlabel("popularity") plt.title("Language Popularity among developers") plt.tight_layout() plt.show()

Show Your Creativity

Automobile Land Speed Records (GR 5-10)

In the first recorded automobile race in 1898, Count Gaston de Chasseloup-Laubat of Paris, France, drove 1 kilometer in 57 seconds for an average speed of 39.2 miles per hour(mph) or 63.1 kilometers per hour (kph). In 1904, Henry Ford drove his Ford Arrow across frozen Lake St.

Clair, MI, at an average speed of 91.4 mph. Now, the North American Eagle is trying to break a land speed record of 800 mph. The Federation International deL'Automobile (FIA), the world's governing body for motor sport and land speed records,recorded the following land speed records.

```
import matplotlib.pyplot as plt
import pandas as pd
```

```
from google.colab import drive
data = pd.read_csv('/content/LandRecords.csv')
```

```
data.head()
```

```
   Speed (mph)          Driver               Car  Engine        Date
0      407.447  Craig Breedlove  Spirit of America  GE J47    8/5/1963
1      413.199        Tom Green   Wingfoot Express  WE J46   10/2/1964
2      434.220        Art Arfons      Green Monster  GE J79   10/5/1964
3      468.719  Craig Breedlove  Spirit of America  GE J79  10/13/1964
4      526.277  Craig Breedlove  Spirit of America  GE J79  10/15/1965
```

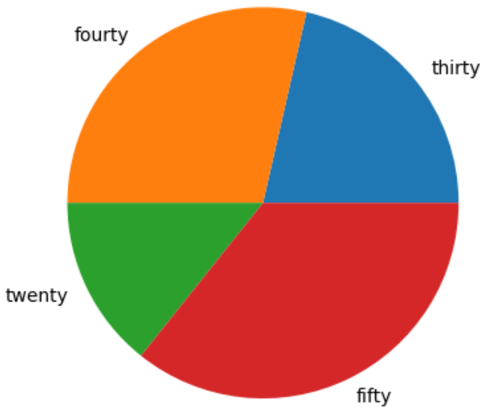
```
from matplotlib import pyplot as plt
```

1. Plotting the Pie Chart

```
slices = [30, 40, 20, 50] #sum needs not be 100plt.pie(slices)
plt.show()
```

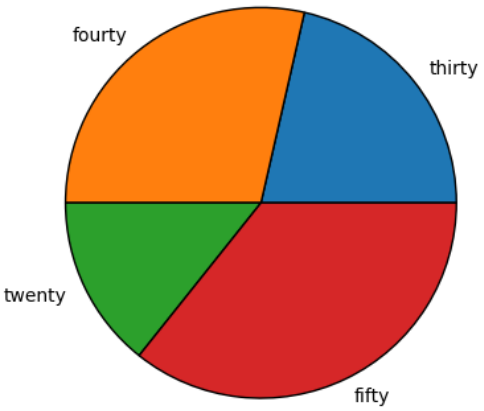
2. Adding labels to the pie chart

```
labels = ['thirty','fourty', 'twenty','fifty']
plt.pie(slices, labels=labels)
plt.show()
```



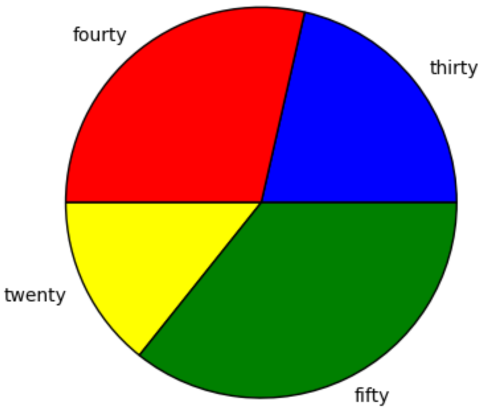
3. setting edge color

```
plt.pie(slices, labels=labels, wedgeprops={'edgecolor':'black'})
plt.show()
```



4. setting color of the slices

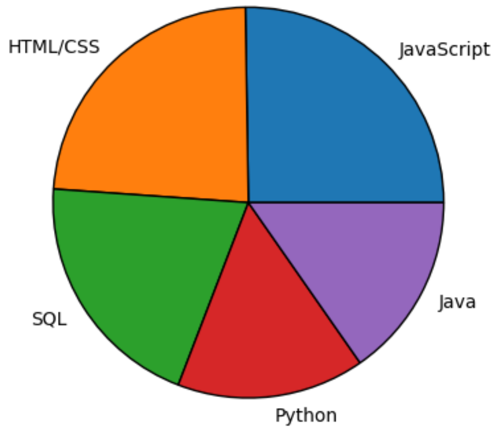
```
color = ['blue','red','yellow','green']

#hexadecimal color codes can also be used
plt.pie(slices, labels=labels, colors=color, wedgeprops={'edgecolor':'black'})
plt.show()
```



5. plotting real world data

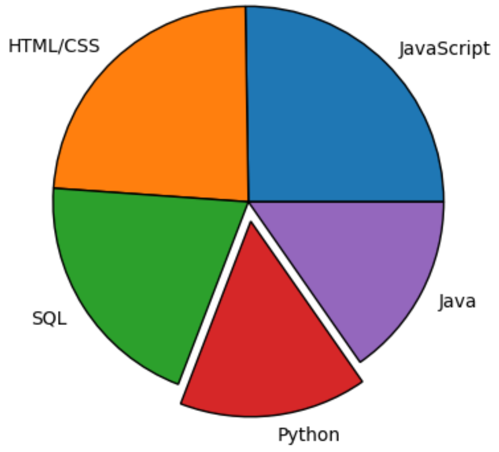
```
labels = ['JavaScript', 'HTML/CSS', 'SQL', 'Python', 'Java']

slices = [59219, 55466, 47544, 36443, 35917]
```

```
plt.pie(slices, labels=labels, wedgeprops={'edgecolor':'black'})
plt.show()
```



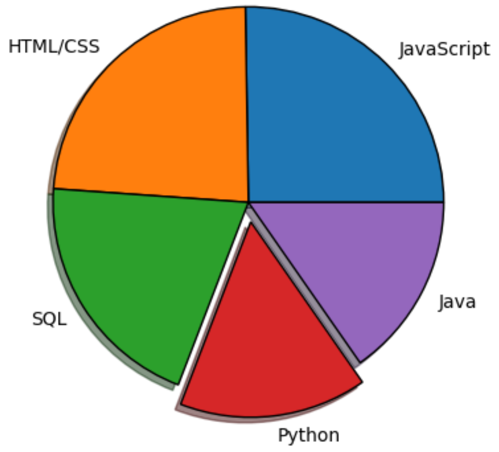
6. Exploding the slice

```
explode = [0, 0, 0, 0.1, 0]

plt.pie(slices, labels=labels, explode=explode, wedgeprops={'edgecolor':'black'})
plt.show()
```



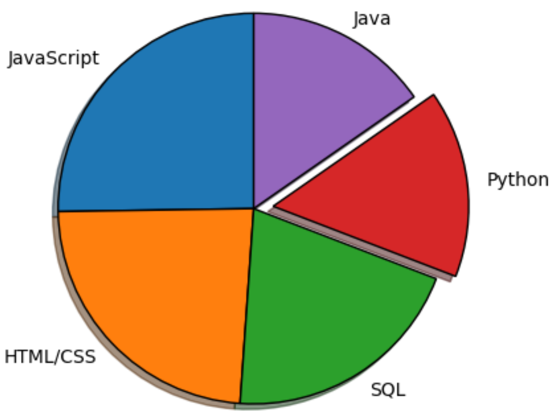
7. adding shadow to the chart

```
plt.pie(slices, labels=labels, explode=explode, shadow=True, wedgeprops={'edgecolor':'black'})
plt.show()
```



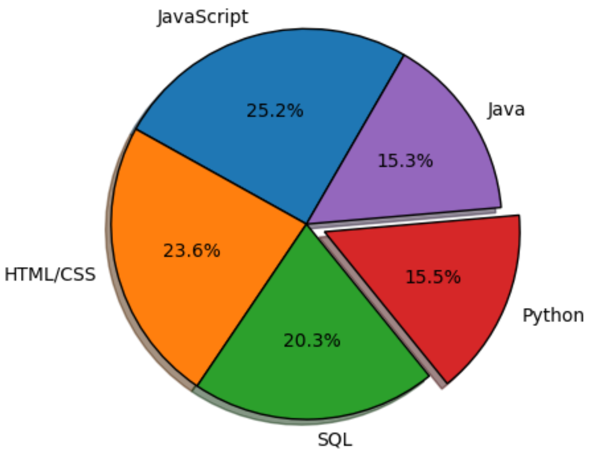
8. setting the starting angle

```
plt.pie(slices, labels=labels, explode=explode, shadow=True, startangle=90, wedgeprops={'edgecolor':'black'})
plt.show()
```



9. displaying percentage of each slices

```
plt.pie(slices, labels=labels, explode=explode, shadow=True, startangle=60, autopct="%0.1f%%", wedgeprops={'edgecolor':'black'})
plt.show()
```



Show Your Creativity

Covid 19 India Data as on 5th Sept 2020

```
import matplotlib.pyplot as plt
import pandas as pd
```

```
from google.colab import drive
data = pd.read_csv('/content/data.csv')
data.head()
```

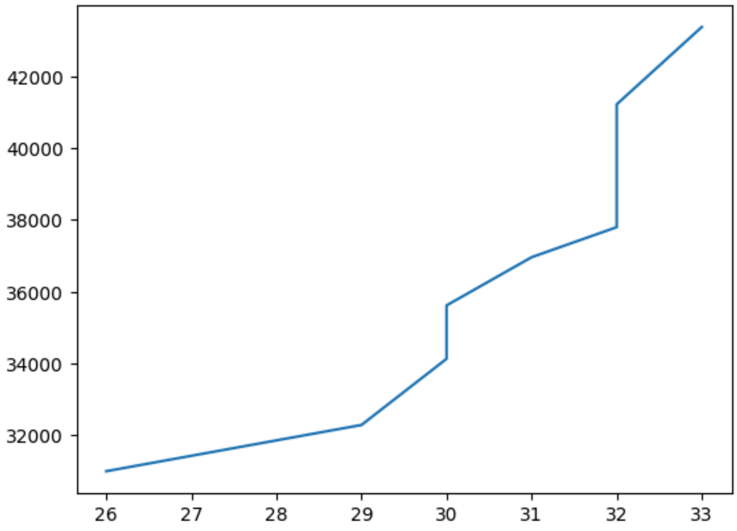
	Responder_id	LanguagesWorkedWith
0	1	HTML/CSS;Java;JavaScript;Python
1	2	C++;HTML/CSS;Python
2	3	HTML/CSS
3	4	C;C++;C#;Python;SQL
4	5	C++;HTML/CSS;Java;JavaScript;Python;SQL;VBA

1. Creating Plots

```
# Installation: pip install matplotlib/ conda install matplotlib
import matplotlib.pyplot as plt
import random
# generating 10 random numbers between 25 to 35
ages = [random.randrange(25,35,1) for ages in range(11)]
ages = sorted(ages, reverse=False)
# generating 10 random numbers between 30k to 45k

devs = [random.randrange(30000,45000,1) for devs in range(11)]
devs = sorted(devs, reverse=False)
```

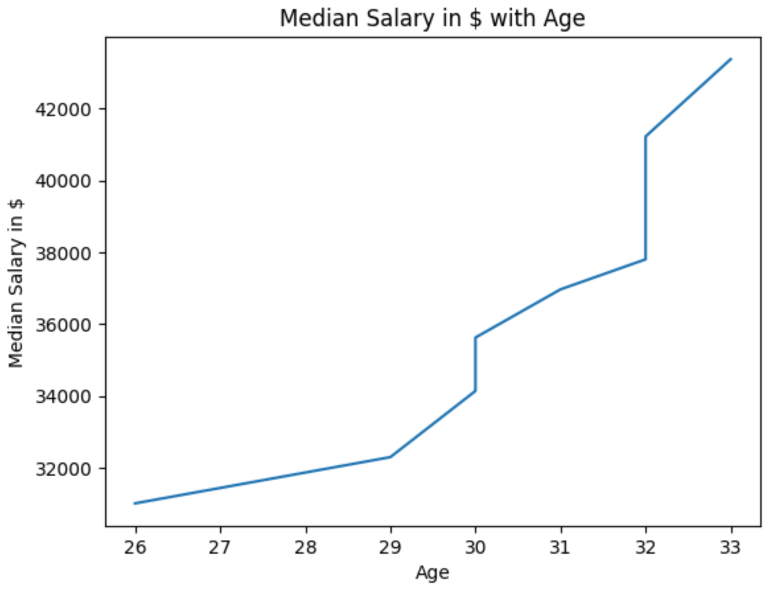
1.1. Plotting Line Plot

```
plt.plot(ages, devs)
plt.show()
```



1.2. Adding title, xlabel and ylabel

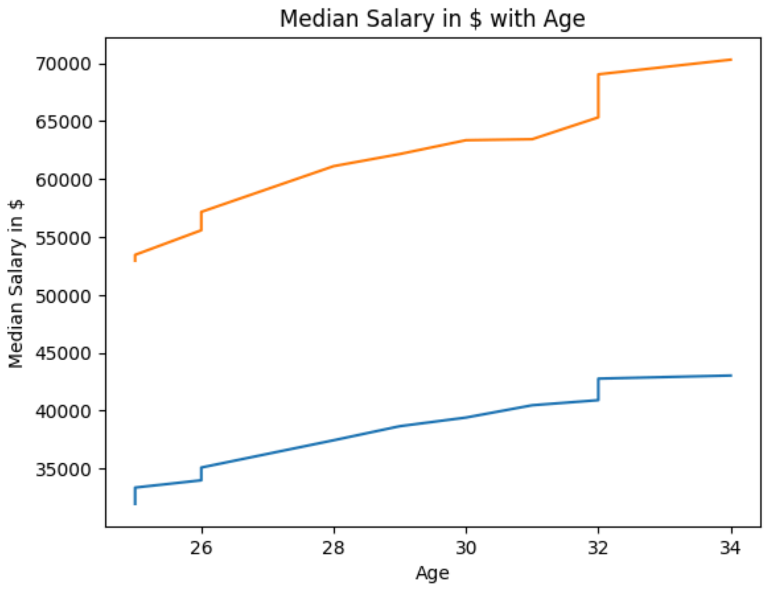
```
plt.plot(ages, devs)
plt.title("Median Salary in $ with Age") # add the title
plt.xlabel("Age") # add xlabel
plt.ylabel("Median Salary in $") #add ylabel
plt.show()
```



1.3. Adding more plot to the same graph

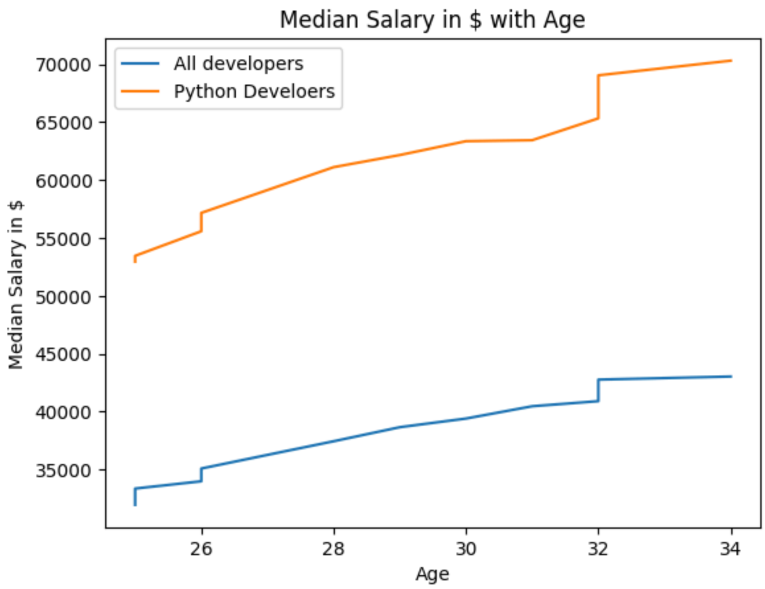
```
#creating 10 random numbers between 50k to 75k
import random
import matplotlib.pyplot as plt
ages = [random.randrange(25,35,1) for ages in range(11)]
ages = sorted(ages, reverse=False)
devs = [random.randrange(30000,45000,1) for devs in range(11)]
devs = sorted(devs, reverse=False)
py_devs = [random.randrange(50000,75000) for py_devs in range(11)]
py_devs = sorted(py_devs, reverse=False)
```

```
plt.plot(ages, devs)
plt.plot(ages, py_devs) # adding other plot to the same figure
plt.title("Median Salary in $ with Age")
plt.xlabel("Age")
plt.ylabel("Median Salary in $")
plt.show()
```



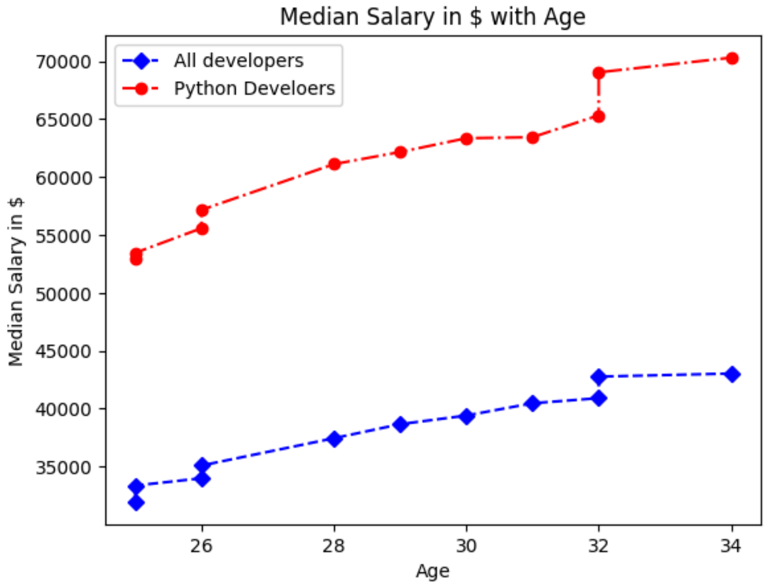
1.4. Adding legend to the plot

```
plt.plot(ages, devs, label = "All developers") # label
plt.plot(ages, py_devs, label = "Python Develoers")
plt.title("Median Salary in $ with Age")
plt.xlabel("Age")
plt.ylabel("Median Salary in $")
plt.legend() #plot the legend
plt.show()
```



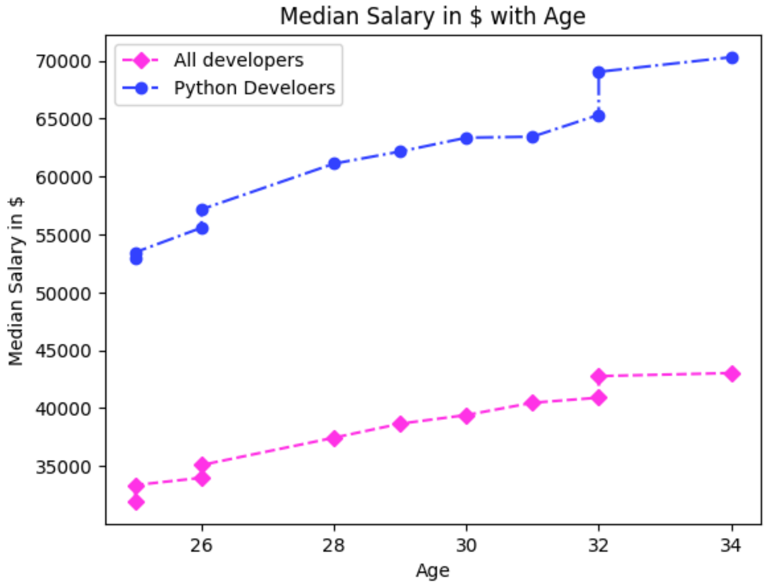
1.5. Setting marker, linestyle and color

```
#https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html

plt.plot(ages, devs, color="blue", linestyle = "--", marker = "D", label = "All developers")
plt.plot(ages, py_devs, color="red", linestyle = "-.", marker = "o", label = "Python Develoers")
plt.title("Median Salary in $ with Age")
plt.xlabel("Age")
plt.ylabel("Median Salary in $")
plt.legend()
plt.show()
```



1.6. Hexadecimal code for colors

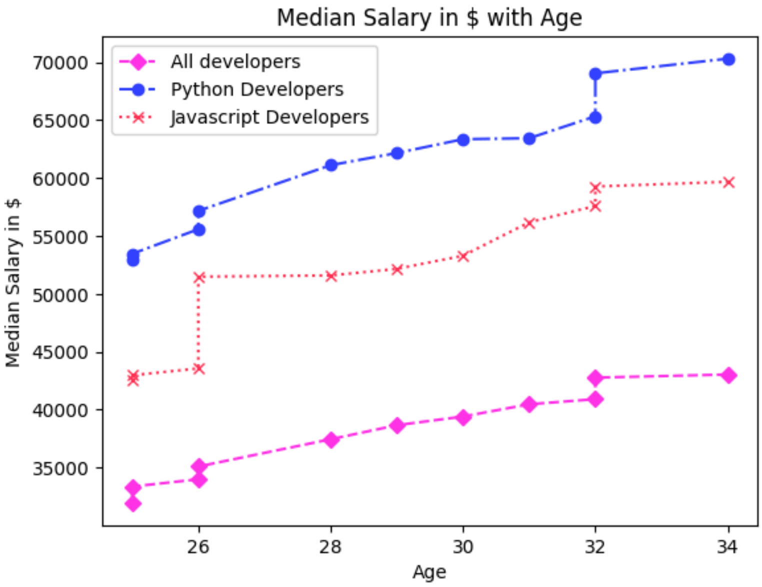
```
#https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html

plt.plot(ages, devs, color="#FF33E9", linestyle = "--", marker = "D", label = "All developers")
plt.plot(ages, py_devs, color="#3344FF", linestyle = "-.", marker = "o", label = "Python Develoers")
plt.title("Median Salary in $ with Age")
plt.xlabel("Age")
plt.ylabel("Median Salary in $")
plt.legend()
plt.show()
```



Adding other plot to the same graph

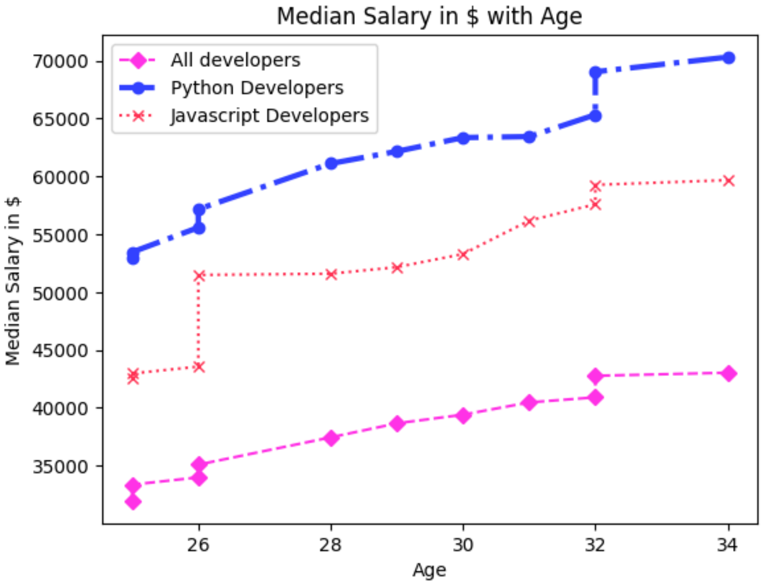
```
#creating 10 random numbers between 40k to 60k

js_devs = [random.randrange(40000,60000) for js_devs in range(11)]
js_devs = sorted(js_devs, reverse=False)
```

```
#https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html

plt.plot(ages, devs, color="#FF33E9", linestyle = "--", marker = "D", label = "All developers")
plt.plot(ages, py_devs, color="#3344FF", linestyle = "-.", marker = "o", label = "Python Developers")
plt.plot(ages, js_devs, color="#FF3355", linestyle = ":", marker = "x", label = "Javascript Developers")
plt.title("Median Salary in $ with Age")
plt.xlabel("Age")
plt.ylabel("Median Salary in $")
plt.legend()
plt.show()
```



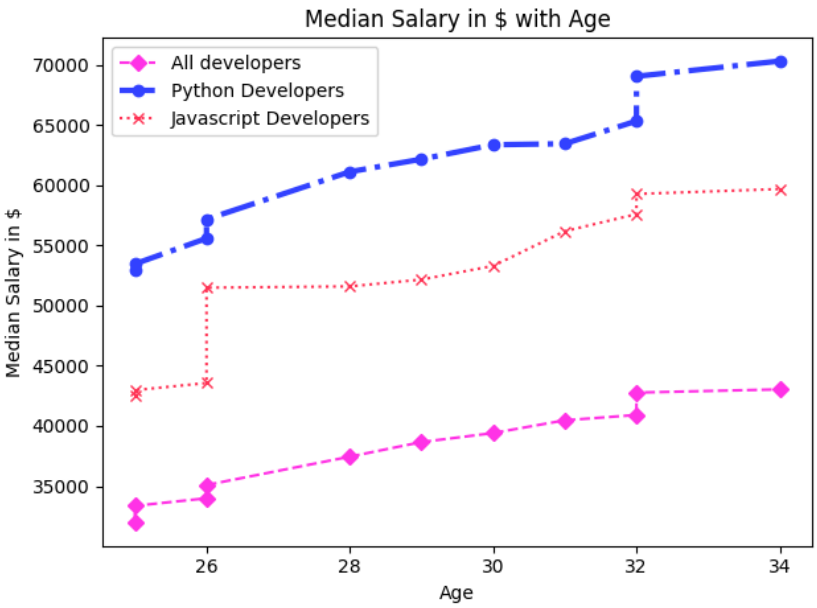
1.7. Changing the line width

```
#https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html

plt.plot(ages, devs, color="#FF33E9", linestyle = "--", marker = "D", label = "All developers")
plt.plot(ages, py_devs, color="#3344FF", linestyle = "-.", marker = "o", linewidth=3, label = "Python Developers")
plt.plot(ages, js_devs, color="#FF3355", linestyle = ":", marker = "x", label = "Javascript Developers")
plt.title("Median Salary in $ with Age")
plt.xlabel("Age")
plt.ylabel("Median Salary in $")
plt.legend()
plt.show()
```



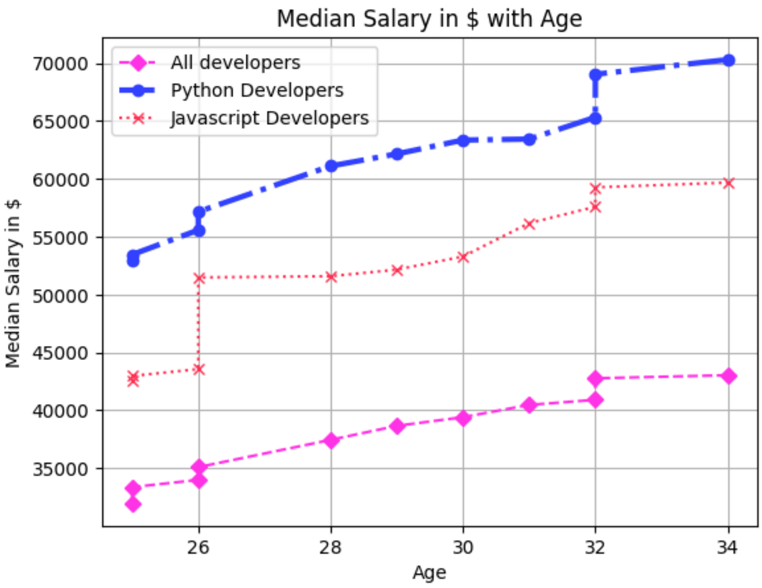
1.8. Add padding to the plot

```
#https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html

plt.plot(ages, devs, color="#FF33E9", linestyle = "--", marker = "D", label = "All developers")
plt.plot(ages, py_devs, color="#3344FF", linestyle = "-.", marker = "o", linewidth=3, label = "Python Developers")
plt.plot(ages, js_devs, color="#FF3355", linestyle = ":", marker = "x", label = "Javascript Developers")
plt.title("Median Salary in $ with Age")
plt.xlabel("Age")
plt.ylabel("Median Salary in $")
plt.legend()
plt.tight_layout() #adds padding
plt.show()
```



1.9. Adding grid to the plot

```
#https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html

plt.plot(ages, devs, color="#FF33E9", linestyle = "--", marker = "D", label = "All developers")
plt.plot(ages, py_devs, color="#3344FF", linestyle = "-.", marker = "o", linewidth=3, label = "Python Developers")
plt.plot(ages, js_devs, color="#FF3355", linestyle = ":", marker = "x", label = "Javascript Developers")
plt.title("Median Salary in $ with Age")
plt.xlabel("Age")
plt.ylabel("Median Salary in $")
plt.grid(True)
plt.legend()
plt.show()
```



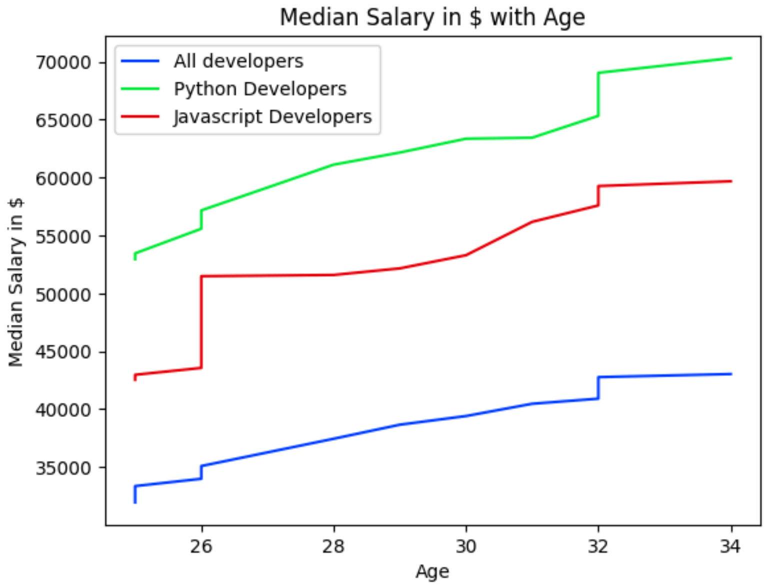
1.10. Changing style of the plot

```
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid', 'bmh', 'classic', 'dark_background', 'fast', 'fivet
```

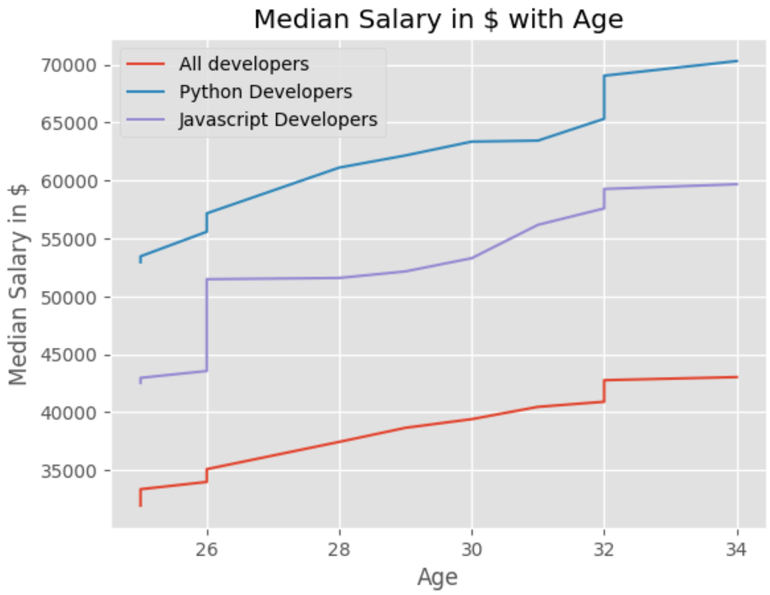
```
#https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html

plt.style.use('seaborn-bright') #to change the style
plt.plot(ages, devs, label = "All developers")
plt.plot(ages, py_devs, label = "Python Developers")
plt.plot(ages, js_devs, label = "Javascript Developers")
plt.title("Median Salary in $ with Age")
plt.xlabel("Age")
plt.ylabel("Median Salary in $")
plt.legend()
plt.show()
```

```
<ipython-input-67-d07ee214d04f>:3: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, a
  plt.style.use('seaborn-bright') #to change the style
```



1.11. Saving the plot

```
#https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html

plt.style.use('ggplot')

plt.plot(ages, devs, label = "All developers")
plt.plot(ages, py_devs, label = "Python Developers")
plt.plot(ages, js_devs, label = "Javascript Developers")

plt.title("Median Salary in $ with Age")
plt.xlabel("Age")
plt.ylabel("Median Salary in $")

plt.legend()

plt.savefig("plot.png")#save the plot

plt.show()
```



for Further Reading click the below link

https://matplotlib.org/tutorials/introductory/pyplot.html

https://pythonbasics.org/matplotlib-line-chart/

```
import matplotlib.pyplot as plt
import pandas as pd
from google.colab import drive
data = pd.read_csv('/content/data_gapminder_gdp_oceania.csv',index_col='country')

print(data)
```

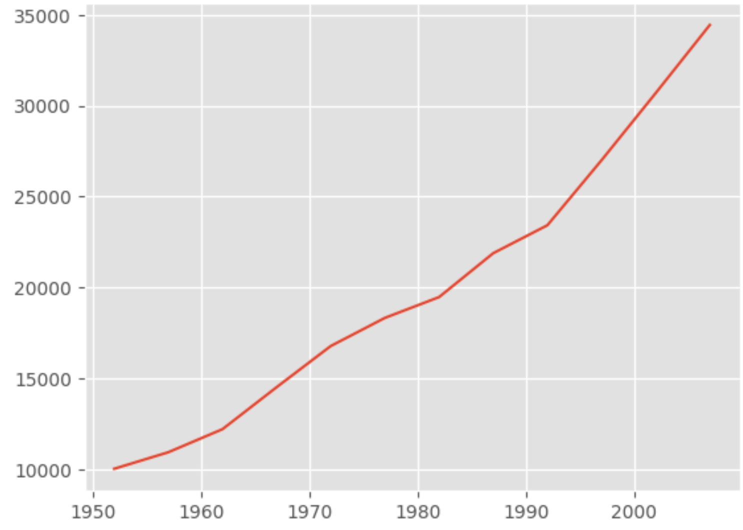
```
             gdpPercap_1952  gdpPercap_1957  gdpPercap_1962  gdpPercap_1967  \
country
Australia       10039.59564     10949.64959     12217.22686     14526.12465
New Zealand     10556.57566     12247.39532     13175.67800     14463.91893

             gdpPercap_1972  gdpPercap_1977  gdpPercap_1982  gdpPercap_1987  \
country
Australia       16788.62948     18334.19751     19477.00928     21888.88903
New Zealand     16046.03728     16233.71770     17632.41040     19007.19129

             gdpPercap_1992  gdpPercap_1997  gdpPercap_2002  gdpPercap_2007
country
Australia       23424.76683     26997.93657     30687.75473     34435.36744
New Zealand     18363.32494     21050.41377     23189.80135     25185.00911
```

Plot data directly from a Pandas dataframe.

- We can also plot Pandas dataframes.
- This implicitly uses matplotlib.pyplot.
- Before plotting, we convert the column headings from a string to integer data type, since they represent numerical values

```
# Extract year from last 4 characters of each column name
# The current column names are structured as 'gdpPercap_(year)',
# so we want to keep the (year) part only for clarity when plotting GDP vs. years
# To do this we use strip(), which removes from the string the characters stated in the argument
# This method works on strings, so we call str before strip()

years = data.columns.str.strip('gdpPercap_')

# Convert year values to integers, saving results back to dataframe

data.columns = years.astype(int)

data.loc['Australia'].plot()
```

```
<Axes: >
```

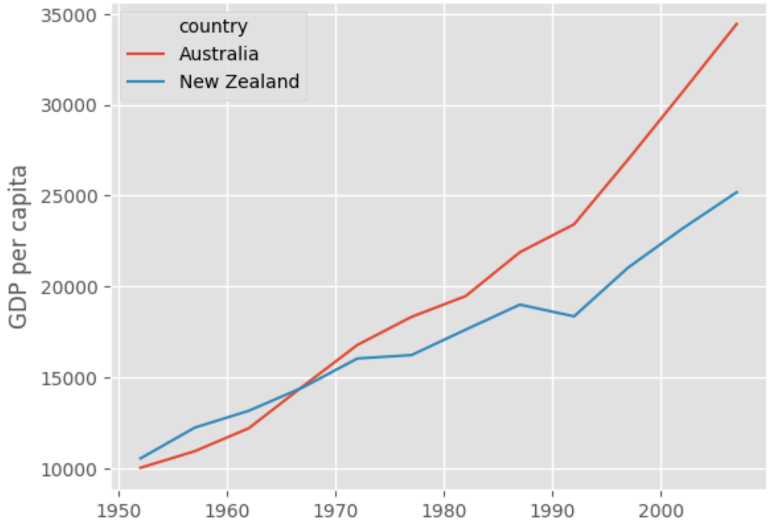


Select and transform data, then plot it.

- By default, DataFrame.plot plots with the rows as the X axis.
- We can transpose the data in order to plot multiple series.

```
data.T.plot()
plt.ylabel('GDP per capita')
```

```
Text(0, 0.5, 'GDP per capita')
```



Data can also be plotted by calling the matplotlib plot function directly.

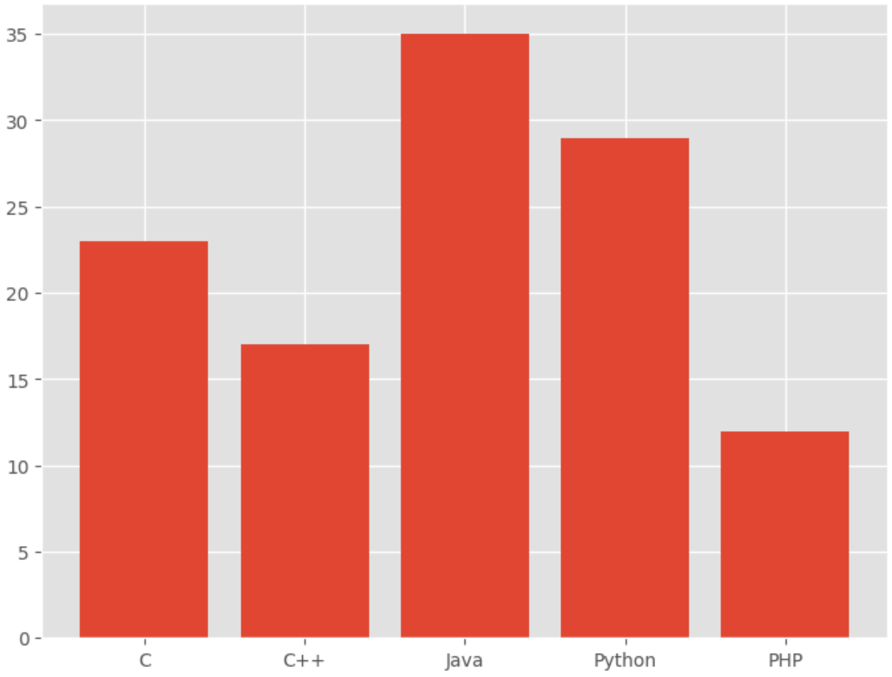
- The command is plt.plot(x, y)
- The color and format of markers can also be specified as an additional optional argument e.g., b- is a blue line, g-- is a green dashed line.

Get Australia data from dataframe

Plot with both countries

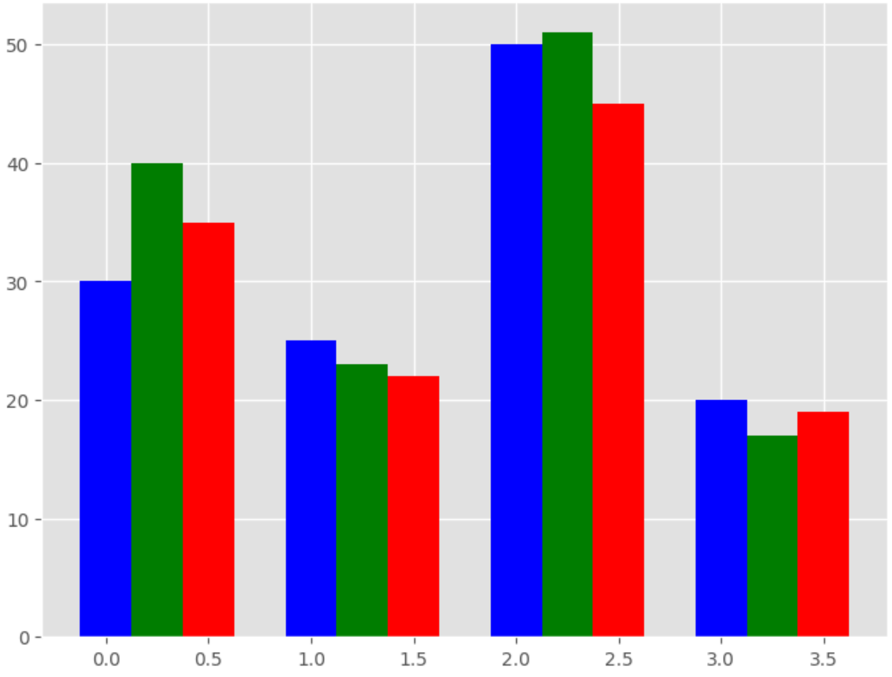
- Select two countries' worth of data.
- Plot with differently-colored markers.
- Create legend.

```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
langs = ['C', 'C++', 'Java', 'Python', 'PHP']
students = [23,17,35,29,12]
ax.bar(langs,students)
plt.show()
```



```
import numpy as np
import matplotlib.pyplot as plt
data = [[30, 25, 50, 20],
[40, 23, 51, 17],
[35, 22, 45, 19]]
X = np.arange(4)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(X + 0.00, data[0], color = 'b', width = 0.25)
ax.bar(X + 0.25, data[1], color = 'g', width = 0.25)
ax.bar(X + 0.50, data[2], color = 'r', width = 0.25)
```

```
<BarContainer object of 4 artists>
```



28-02-2024

```
import numpy as np
import pandas  as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
```

```
breast = load_breast_cancer()
breast_data = breast.data
print(breast_data)
print(breast_data.shape)
```

```
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
(569, 30)
```

```
cancer = load_breast_cancer(as_frame=True)
```

```
df=cancer.frame
```

```
df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	wors are
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.

5 rows × 31 columns

```
df.shape
```

```
(569, 31)
```

```
breast_labels = breast.target
print(breast_labels)
print(breast_labels.shape)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
(569,)
```

```
labels = np.reshape(breast_labels,(569,1))
final_breast_data = np.concatenate([breast_data,labels],axis=1)
print(final_breast_data.shape)
```

```
(569, 31)
```

```
breast_dataset = pd.DataFrame(final_breast_data)
print(breast_dataset.head())
```

```
      0      1       2       3        4        5       6        7       8   \
0  17.99  10.38  122.80  1001.0  0.11840  0.27760  0.3001  0.14710  0.2419
1  20.57  17.77  132.90  1326.0  0.08474  0.07864  0.0869  0.07017  0.1812
2  19.69  21.25  130.00  1203.0  0.10960  0.15990  0.1974  0.12790  0.2069
3  11.42  20.38   77.58   386.1  0.14250  0.28390  0.2414  0.10520  0.2597
```

```
4  20.29  14.34  135.10  1297.0  0.10030  0.13280  0.1980  0.10430  0.1809

         9   ...     21      22      23      24      25      26      27  \
0  0.07871  ...  17.33  184.60  2019.0  0.1622  0.6656  0.7119  0.2654
1  0.05667  ...  23.41  158.80  1956.0  0.1238  0.1866  0.2416  0.1860
2  0.05999  ...  25.53  152.50  1709.0  0.1444  0.4245  0.4504  0.2430
3  0.09744  ...  26.50   98.87   567.7  0.2098  0.8663  0.6869  0.2575
4  0.05883  ...  16.67  152.20  1575.0  0.1374  0.2050  0.4000  0.1625

       28       29   30
0  0.4601  0.11890  0.0
1  0.2750  0.08902  0.0
2  0.3613  0.08758  0.0
3  0.6638  0.17300  0.0
4  0.2364  0.07678  0.0

[5 rows x 31 columns]
```

```
features = breast.feature_names
print(features)
```

```
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
```

```
features_labels = np.append(features,'label')
```

```
breast_dataset.columns = features_labels
breast_dataset.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	wors are
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.

5 rows × 31 columns

```
breast_dataset['label'].replace(0,'benign',inplace = True)
breast_dataset['label'].replace(1,'malignant',inplace = True)
breast_dataset.tail()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	wo a
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10	202
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00	173
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.70	112
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.60	182
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	30.37	59.16	26

5 rows × 31 columns

```
from sklearn.preprocessing import StandardScaler
x = breast_dataset.loc[:, features].values
x = StandardScaler().fit_transform(x) # normalizing the features
print(x.shape)
```

```
(569, 30)
```

```
np.mean(x),np.std(x)
```

```
(-6.118909323768877e-16, 1.0)
```

```
feat_cols = ['features'+str(1) for i in range (x.shape[1])]
normalised_breast = pd.DataFrame(x,columns = feat_cols)
print(normalised_breast)
```

```
     features1  features1  features1  features1  features1  features1  \
0     1.097064  -2.073335   1.269934   0.984375   1.568466   3.283515   
1     1.829821  -0.353632   1.685955   1.908708  -0.826962  -0.487072   
2     1.579888   0.456187   1.566503   1.558884   0.942210   1.052926   
3    -0.768909   0.253732  -0.592687  -0.764464   3.283553   3.402909   
4     1.750297  -1.151816   1.776573   1.826229   0.280372   0.539340   
..         ...        ...        ...        ...        ...        ...   
564   2.110995   0.721473   2.060786   2.343856   1.041842   0.219060   
565   1.704854   2.085134   1.615931   1.723842   0.102458  -0.017833   
566   0.702284   2.045574   0.672676   0.577953  -0.840484  -0.038680   
567   1.838341   2.336457   1.982524   1.735218   1.525767   3.272144   
568  -1.808401   1.221792  -1.814389  -1.347789  -3.112085  -1.150752   

     features1  features1  features1  features1  ...  features1  features1  \
0     2.652874   2.532475   2.217515   2.255747  ...   1.886690  -1.359293   
1    -0.023846   0.548144   0.001392  -0.868652  ...   1.805927  -0.369203   
2     1.363478   2.037231   0.939685  -0.398008  ...   1.511870  -0.023974   
3     1.915897   1.451707   2.867383   4.910919  ...  -0.281464   0.133984   
4     1.371011   1.428493  -0.009560  -0.562450  ...   1.298575  -1.466770   
..         ...        ...        ...        ...  ...        ...        ...   
564   1.947285   2.320965  -0.312589  -0.931027  ...   1.901185   0.117700   
565   0.693043   1.263669  -0.217664  -1.058611  ...   1.536720   2.047399   
566   0.046588   0.105777  -0.809117  -0.895587  ...   0.561361   1.374854   
567   3.296944   2.658866   2.137194   1.043695  ...   1.961239   2.237926   
568  -1.114873  -1.261820  -0.820070  -0.561032  ...  -1.410893   0.764190   

     features1  features1  features1  features1  features1  features1  \
0     2.303601   2.001237   1.307686   2.616665   2.109526   2.296076   
1     1.535126   1.890489  -0.375612  -0.430444  -0.146749   1.087084   
2     1.347475   1.456285   0.527407   1.082932   0.854974   1.955000   
3    -0.249939  -0.550021   3.394275   3.893397   1.989588   2.175786   
4     1.338539   1.220724   0.220556  -0.313395   0.613179   0.729259   
..         ...        ...        ...        ...        ...        ...   
564   1.752563   2.015301   0.378365  -0.273318   0.664512   1.629151   
565   1.421940   1.494959  -0.691230  -0.394820   0.236573   0.733827   
566   0.579001   0.427906  -0.809587   0.350735   0.326767   0.414069   
567   2.303601   1.653171   1.430427   3.904848   3.197605   2.289985   
568  -1.432735  -1.075813  -1.859019  -1.207552  -1.305831  -1.745063   

     features1  features1  
0     2.750622   1.937015  
1    -0.243890   0.281190  
2     1.152255   0.201391  
3     6.046041   4.935010  
4    -0.868353  -0.397100  
..         ...        ...  
564  -1.360158  -0.709091  
565  -0.531855  -0.973978  
566  -1.104549  -0.318409  
567   1.919083   2.219635  
568  -0.048138  -0.751207  

[569 rows x 30 columns]
```

```
normalised_breast.head()
```

	features1	features1	features1	features1	features1	features1	features1	featur
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.532
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.548
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.037
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897	1.451
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011	1.428

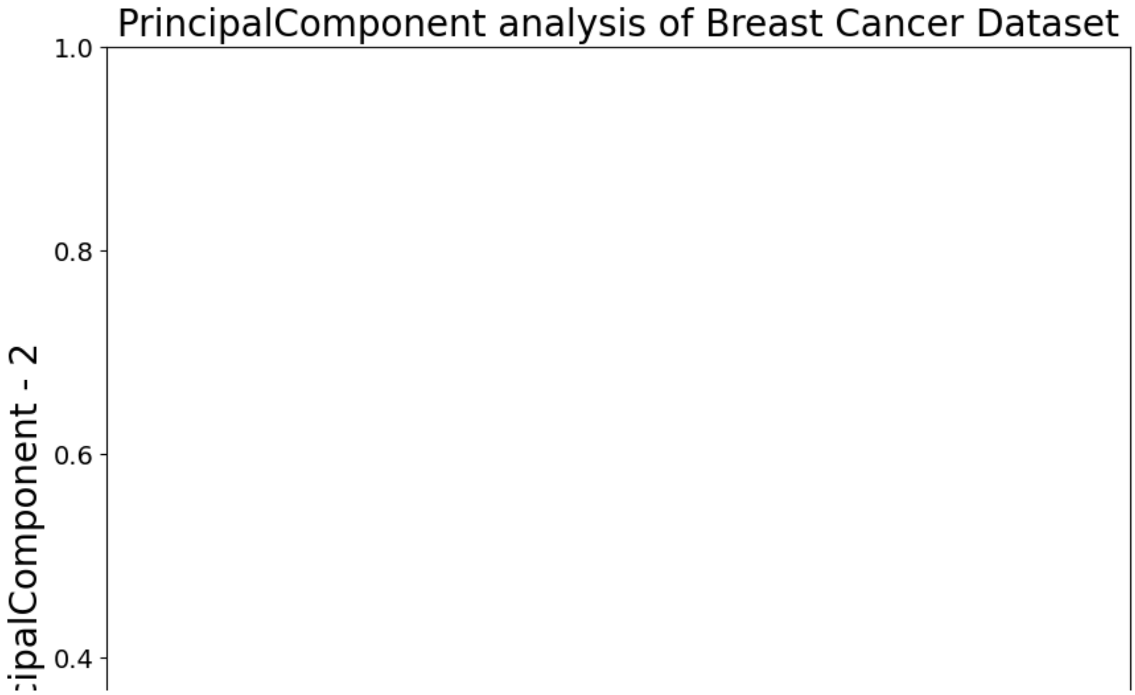
5 rows × 30 columns

```
from sklearn.decomposition import PCA
pca_breast = PCA(n_components=2)
PrincipalComponents_breast = pca_breast.fit_transform(x)
Principal_breast_Df = pd.DataFrame(data = PrincipalComponents_breast,columns = ['principal component 1','principal component 2'])
Principal_breast_Df.tail()
```

	principal component 1	principal component 2
564	6.439315	-3.576817
565	3.793382	-3.584048
566	1.256179	-1.902297
567	10.374794	1.672010
568	-5.475243	-0.670637

```
import matplotlib.pyplot as plt
plt.figure()
plt.figure(figsize = (10,10))
plt.xticks(fontsize =12)
plt.yticks(fontsize = 14)
plt.xlabel('PrincipalComponent - 1',fontsize=20)
plt.ylabel('PrincipalComponent - 2',fontsize=20)
plt.title("PrincipalComponent analysis of Breast Cancer Dataset",fontsize=20)
targets = ['Benign','Malignant']
colors = ['r','g']
for target,color in zip(targets,colors):
  indicesTokeep = breast_dataset['label'] == target
  plt.scatter(Principal_breast_Df.loc[indicesTokeep,'Principal component 1']
             ,Principal_breast_Df.loc[indicesTokeep,'Principal component 2'],c = color,s=5)

  plt.legend(targets,prop={'size':15})
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3801             try:
-> 3802                 return self._engine.get_loc(casted_key)
   3803             except KeyError as err:

                                 10 frames
pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Principal component 1'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3802                 return self._engine.get_loc(casted_key)
   3803             except KeyError as err:
-> 3804                 raise KeyError(key) from err
   3805             except TypeError:
   3806                 # If we have a listlike key, _check_indexing_error will raise

KeyError: 'Principal component 1'

<Figure size 640x480 with 0 Axes>
```



2.3 correlation Regression

```
import matplotlib.pyplot as plt
import seaborn as sns
df = sns.load_dataset('iris')
#without regression
sns.pairplot(df,kind="scatter")
plt.show()
```