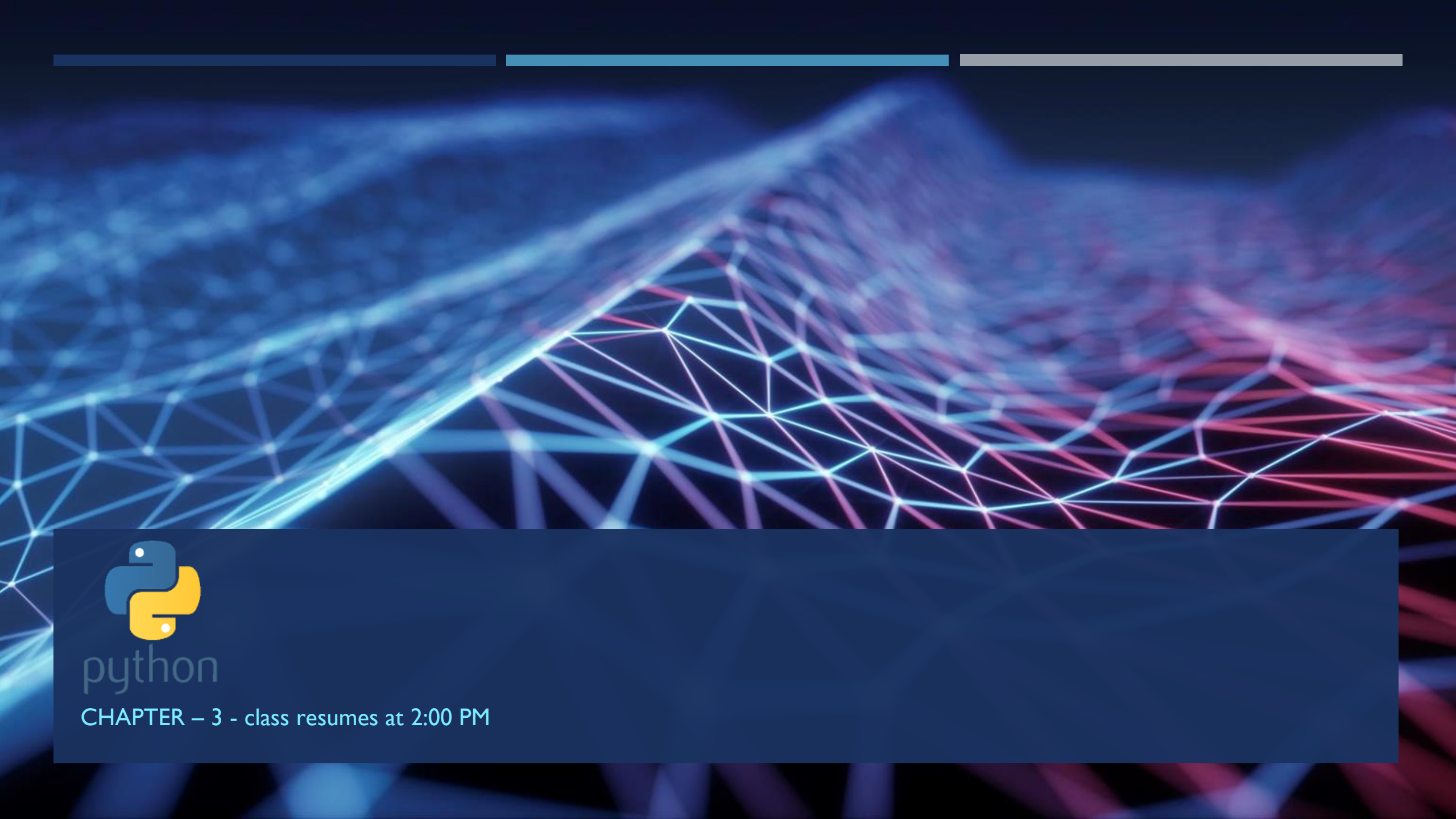
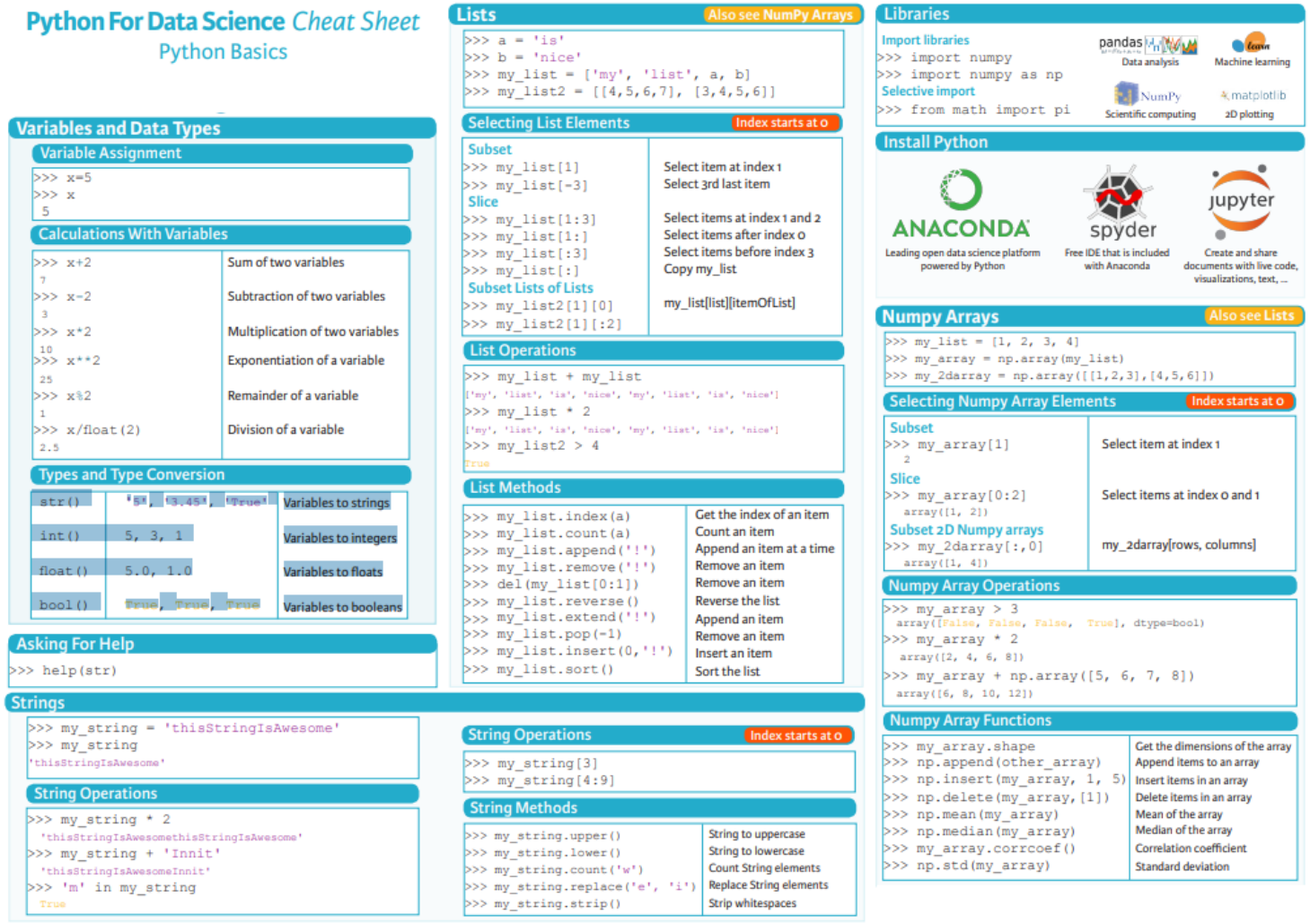


CHAPTER – 3 - class resumes at 2:00 PM

RECAP



Python For Data Science *Cheat Sheet*

Python Basics

Variables and Data Types

Variable Assignment

```
>>> x=5
>>> x
5
```

Calculations With Variables

```
>>> x+2
7
>>> x-2
3
>>> x*2
10
>>> x**2
25
>>> x%2
1
>>> x/float(2)
2.5
```

Sum of two variables
Subtraction of two variables
Multiplication of two variables
Exponentiation of a variable
Remainder of a variable
Division of a variable

Types and Type Conversion

```
str()     '5', '3.45', 'True'     Variables to strings
int()     5, 3, 1                 Variables to integers
float()   5.0, 1.0                Variables to floats
bool()    True, True, True        Variables to booleans
```

Asking For Help

```
>>> help(str)
```

Lists

Also see NumPy Arrays

```
>>> a = 'is'
>>> b = 'nice'
>>> my_list = ['my', 'list', a, b]
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

Selecting List Elements

Index starts at 0

Subset

```
>>> my_list[1]
>>> my_list[-3]
```

Select item at index 1
Select 3rd last item

Slice

```
>>> my_list[1:3]
>>> my_list[1:]
>>> my_list[:3]
>>> my_list[:]
```

Select items at index 1 and 2
Select items after index 0
Select items before index 3
Copy my_list

Subset Lists of Lists

```
>>> my_list2[1][0]
>>> my_list2[1][:2]
```

my_list[list][itemOfList]

List Operations

```
>>> my_list + my_list
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list * 2
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list2 > 4
True
```

List Methods

```
>>> my_list.index(a)
>>> my_list.count(a)
>>> my_list.append('!')
>>> my_list.remove('!')
>>> del(my_list[0:1])
>>> my_list.reverse()
>>> my_list.extend('!')
>>> my_list.pop(-1)
>>> my_list.insert(0,'!')
>>> my_list.sort()
```

Get the index of an item
Count an item
Append an item at a time
Remove an item
Remove an item
Reverse the list
Append an item
Remove an item
Insert an item
Sort the list

Libraries

Import libraries

```
>>> import numpy
>>> import numpy as np
```

Selective import

```
>>> from math import pi
```

pandas — Data analysis
Machine learning
NumPy — Scientific computing
matplotlib — 2D plotting

Install Python

ANACONDA — Leading open data science platform powered by Python
spyder — Free IDE that is included with Anaconda
jupyter — Create and share documents with live code, visualizations, text, ...

Numpy Arrays

Also see Lists

```
>>> my_list = [1, 2, 3, 4]
>>> my_array = np.array(my_list)
>>> my_2darray = np.array([[1,2,3],[4,5,6]])
```

Selecting Numpy Array Elements

Index starts at 0

Subset

```
>>> my_array[1]
2
```

Select item at index 1

Slice

```
>>> my_array[0:2]
array([1, 2])
```

Select items at index 0 and 1

Subset 2D Numpy arrays

```
>>> my_2darray[:,0]
array([1, 4])
```

my_2darray[rows, columns]

Numpy Array Operations

```
>>> my_array > 3
array([False, False, False,  True], dtype=bool)
>>> my_array * 2
array([2, 4, 6, 8])
>>> my_array + np.array([5, 6, 7, 8])
array([6, 8, 10, 12])
```

Numpy Array Functions

```
>>> my_array.shape
>>> np.append(other_array)
>>> np.insert(my_array, 1, 5)
>>> np.delete(my_array,[1])
>>> np.mean(my_array)
>>> np.median(my_array)
>>> my_array.corrcoef()
>>> np.std(my_array)
```

Get the dimensions of the array
Append items to an array
Insert items in an array
Delete items in an array
Mean of the array
Median of the array
Correlation coefficient
Standard deviation

Strings

```
>>> my_string = 'thisStringIsAwesome'
>>> my_string
'thisStringIsAwesome'
```

String Operations

```
>>> my_string * 2
'thisStringIsAwesomethisStringIsAwesome'
>>> my_string + 'Innit'
'thisStringIsAwesomeInnit'
>>> 'm' in my_string
True
```

String Operations

Index starts at 0

```
>>> my_string[3]
>>> my_string[4:9]
```

String Methods

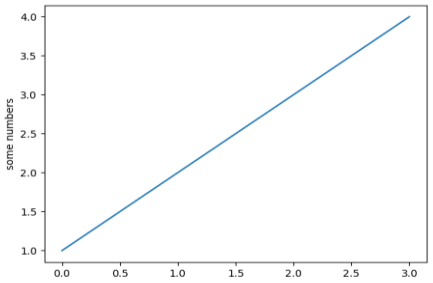
```
>>> my_string.upper()
>>> my_string.lower()
>>> my_string.count('w')
>>> my_string.replace('e', 'i')
>>> my_string.strip()
```

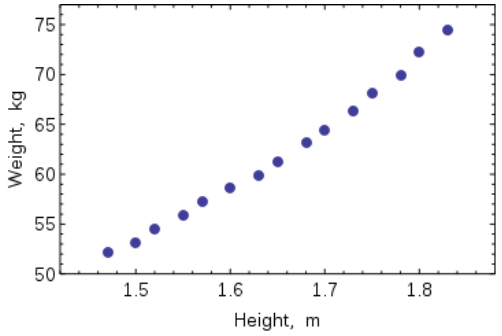
String to uppercase
String to lowercase
Count String elements
Replace String elements
Strip whitespaces

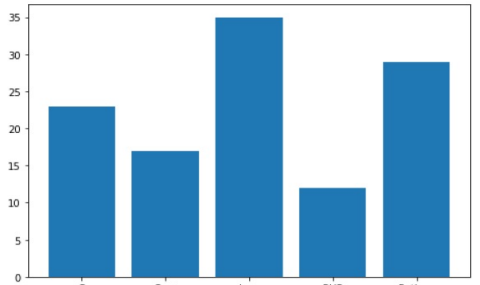
DATA VISULAZATION IN PYTHON

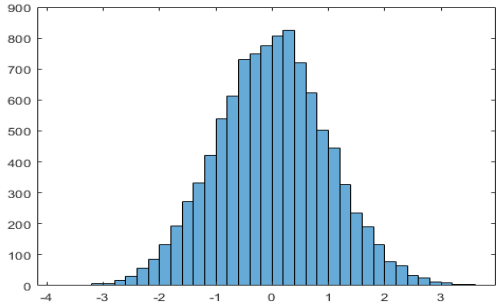
MATPLOTLIB

- Matplotlib is an amazing visualization library in Python for 2D plots of arrays.
- Matplotlib is a multi-platform data visualization library
- One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals.
- Matplotlib consists of several plots like line, scatter, bar, histogram etc.









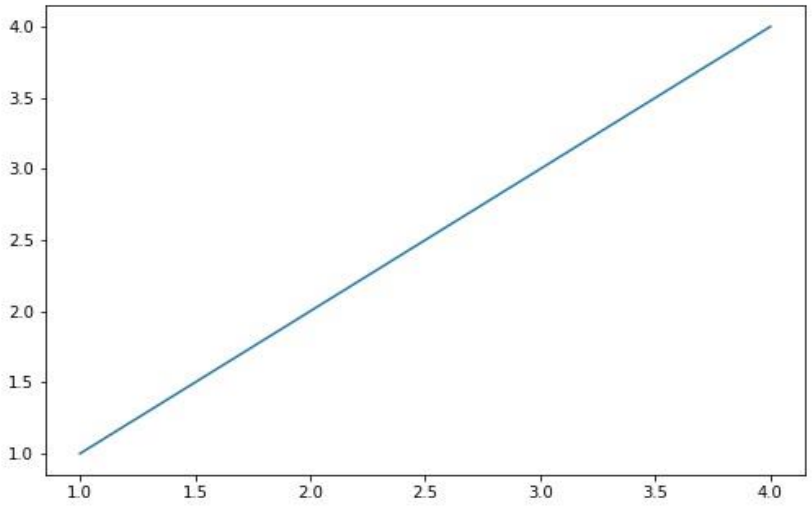
IMPORT MATPLOTLIB

Before you can start, you should import matplotlib.pyplot as plt. pyplot is a sub-package of matplotlib.

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
```

LINE PLOT

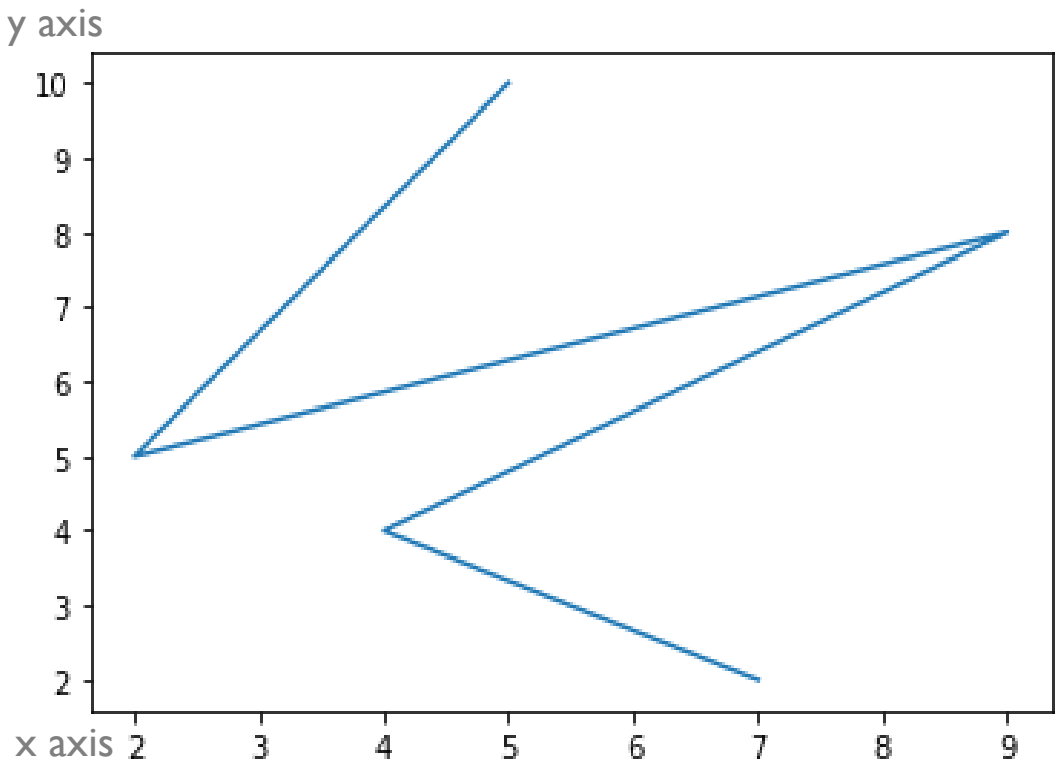
- A Line plot can be defined as a graph that displays data as points or check marks above a number line, showing the frequency of each value.
- A line plot is a graph that shows frequency of data along a number line.
- To show the graph we'll use plt.plot(x,y), as you can see the first value represents the x axis & the second value represents the y axis.



LINE PLOT - EXAMPLE

#Create line plot of using X & Y values

```
# importing matplotlib module
import matplotlib.pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7]
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot
plt.plot(x,y)
# function to show the plot
plt.show() i
```



RANGE FUNCTION

The range() is a built-in function of Python which returns a range object, which is nothing but a sequence of integers. i.e., Python range() generates the integer numbers between the given start integer to the stop integer, which is generally used to iterate over with for loop.

Using range function to generate a list from 1 to 10

```
#create a list using range function
number_list = list(range(1, 10))
#print list
print(number_list)
```

Out: [1, 2, 3, 4, 5, 6, 7, 8, 9]

TASK

Create a list using the range() function starting from year 1950 to 2100.

ANSWER

```
#create a list using range function
year = list(range(1950, 2101))
#print list
print(year)
```

TASK

Create a list using the range() function starting from year 1950 to 2101 with the intervals of 10.

Your answer should look like: [1950, 1960, 1970, 1980, 1990, 2000, 2010, 2020, 2030, 2040, 2050, 2060, 2070, 2080, 2090, 2100]

ANSWER

```
#create a list using range function
year = list(range(1950, 2101, 10))
#print list
print(year)
```

TASK

Create a neg_list using the range() function starting from -1 to -100.

ANSWER

```
#create a list using range function
neg_list = list(range(-101, -1))
#print list
print(neg_list)
```

WORLD POPULATION – CASE STUDY

The world population has grown over the past years. Will it continue to do so? The world bank has estimates of the world population for the years1950 up to 2100.

TASK

Use the list you created for year 1950 to 2101 print() the last item from both the year and the pop list (population forecast in billions) to see what the predicted population for the year 2100 is.

use this pop list:

pop=[2.53,2.57,2.62,2.67,2.71,2.76,2.81,2.86,2.92,2.97,3.03,3.08,3.14,3.2,3.26,3.33,3.4,3.47,3.54,3.62,3.69,3.77,3.84,3.92,4.,4.07,4.15,4.22,4.3,4.37,4.45,4.53,4.61,4.69,4.78,4.86,4.95,5.05,5.14,5.23,5.32,5.41,5.49,5.58,5.66,5.74,5.82,5.9,5.98,6.05,6.13,6.2,6.28,6.36,6.44,6.51,6.59,6.67,6.75,6.83,6.92,7.,7.08,7.16,7.24,7.32,7.4,7.48,7.56,7.64,7.72,7.79,7.87,7.94,8.01,8.08,8.15,8.22,8.29,8.36,8.42,8.49,8.56,8.62,8.68,8.74,8.8,8.86,8.92,8.98,9.04,9.09,9.15,9.2,9.26,9.31,9.36,9.41,9.46,9.5,9.55,9.6,9.64,9.68,9.73,9.77,9.81,9.85,9.88,9.92,9.96,9.99,10.03,10.06,10.09,10.13,10.16,10.19,10.22,10.25,10.28,10.31,10.33,10.36,10.38,10.41,10.43,10.46,10.48,10.5,10.52,10.55,10.57,10.59,10.61,10.63,10.65,10.66,10.68,10.7,10.72,10.73,10.75,10.77,10.78,10.79,10.81,10.82,10.83,10.84,10.85]

ANSWER

```
#create a list using range function
year = list(range(1950, 2101))
pop =
[2.53,2.57,2.62,2.67,2.71,2.76,2.81,2.86,2.92,2.97,3.03,3.08,3.14,3.2,3.26,3.33,3.4,3.47,3.54,3.62,3.69,3.77,3.84,3.92,4.,4.07,4.15,
4.22,4.3,4.37,4.45,4.53,4.61,4.69,4.78,4.86,4.95,5.05,5.14,5.23,5.32,5.41,5.49,5.58,5.66,5.74,5.82,5.9,5.98,6.05,6.13,6.2,6.28,6.36,
6.44,6.51,6.59,6.67,6.75,6.83,6.92,7.,7.08,7.16,7.24,7.32,7.4,7.48,7.56,7.64,7.72,7.79,7.87,7.94,8.01,8.08,8.15,8.22,8.29,8.36,8.42,
8.49,8.56,8.62,8.68,8.74,8.8,8.86,8.92,8.98,9.04,9.09,9.15,9.2,9.26,9.31,9.36,9.41,9.46,9.5,9.55,9.6,9.64,9.68,9.73,9.77,9.81,9.85,9.
88,9.92,9.96,9.99,10.03,10.06,10.09,10.13,10.16,10.19,10.22,10.25,10.28,10.31,10.33,10.36,10.38,10.41,10.43,10.46,10.48,10.5,1
0.52,10.55,10.57,10.59,10.61,10.63,10.65,10.66,10.68,10.7,10.72,10.73,10.75,10.77,10.78,10.79,10.81,10.82,10.83,10.84,10.85]


# Print the last item from year and pop
print(year[-1])
print(pop[-1])
```

TASK

Use the list you created for year 1950 to 2101 print()

Create line plot of world projected population. Year should be mapped on the horizontal axis, population on the vertical axis.

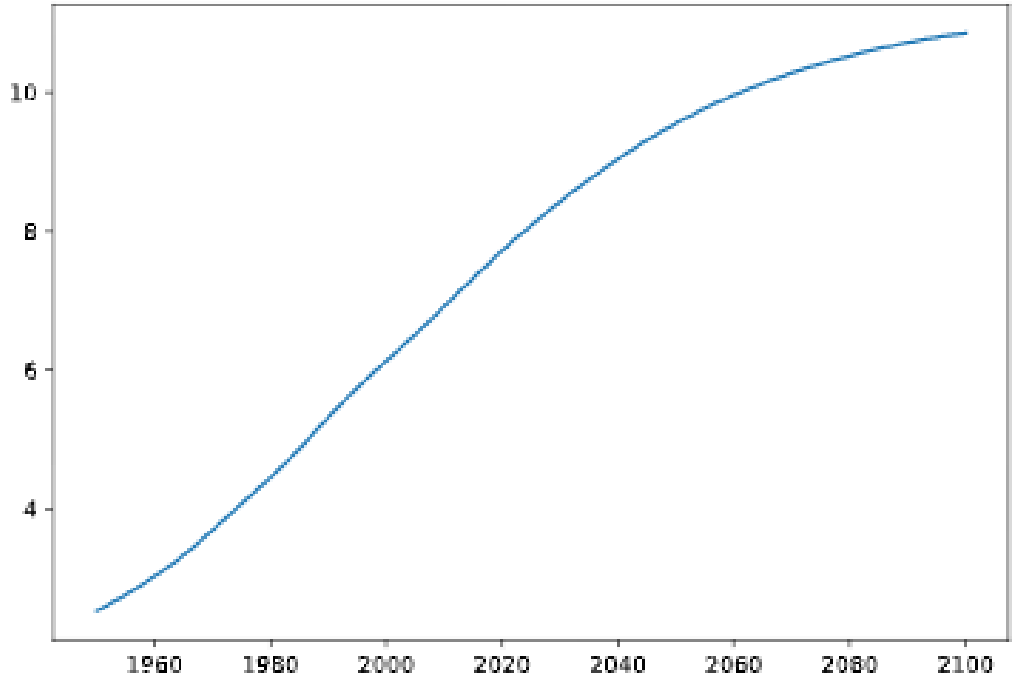
ANSWER

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
#make line plot  year on x-axis and pop on y-axis
plt.plot(year,pop)
#display the plot
plt.show()
```

LINE PLOT

Have another look at the plot you created in the previous task, shown below.

Based on the plot, in approximately what year will there be more than ten billion human beings on this planet?



TASK

Create a line plot using the following data for monthly savings.

Month	Savings $
Jan	209.00
Feb	250.00
Mar	209.00
Apr	297.00
May	356.00
Jun	388.00
Jul	305.00
Aug	397.00
Sep	362.00
Oct	237.00
Nov	282.00
Dec	238.00

ANSWER

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
month=['Jen','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nev','Dec']
saving=[209.00,250.00,209.00,297.00,356.00,388.00,305.00,397.00,362.00,237.00,282.00,238.00]
#make line plot  year on x-axis and pop on y-axis
plt.plot(month,saving)
#display the plot
plt.show()
```

WORLD GDP PER CAPITA – TASK BACKGROUND

We've collected data life_exp which contains the life expectancy for each country and gdp_cap, which contains the GDP per capita (i.e. per person) for each country expressed in US Dollars.

GDP stands for Gross Domestic Product. It basically represents the size of the economy of a country. Divide this by the population and we get the GDP per capita

TASK

Print the last item from both the list (GDP per Capita) gdp_cap, and the list (Life Expectancy) life_exp.

For completing this task use code below.

```
 import pandas as pd
import numpy as np
country_Data = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/gapminder.csv')
np_country_Data=np.array(country_Data)
gdp_cap=np_country_Data[:,6]
life_exp = np_country_Data[:,5]
#gdp_cap, which contains the GDP per capita (i.e. per person) for each country expressed in US Dollars.
#life_exp which contains the life expectancy for each country
```

ANSWER

```
 import pandas as pd
import numpy as np
country_Data=pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/gapminder.csv'
)
#gdp_cap, which contains the GDP per capita (i.e. per person) for each country expressed in US Dollars.
np_country_Data=np.array(country_Data)
gdp_cap=np_country_Data[:,6]
#life_exp which contains the life expectancy for each country
life_exp = np_country_Data[:,5]
  #print out the list item of life_exp and gdp_cap
print(life_exp[-1])
print(gdp_cap[-1])
```

TASK

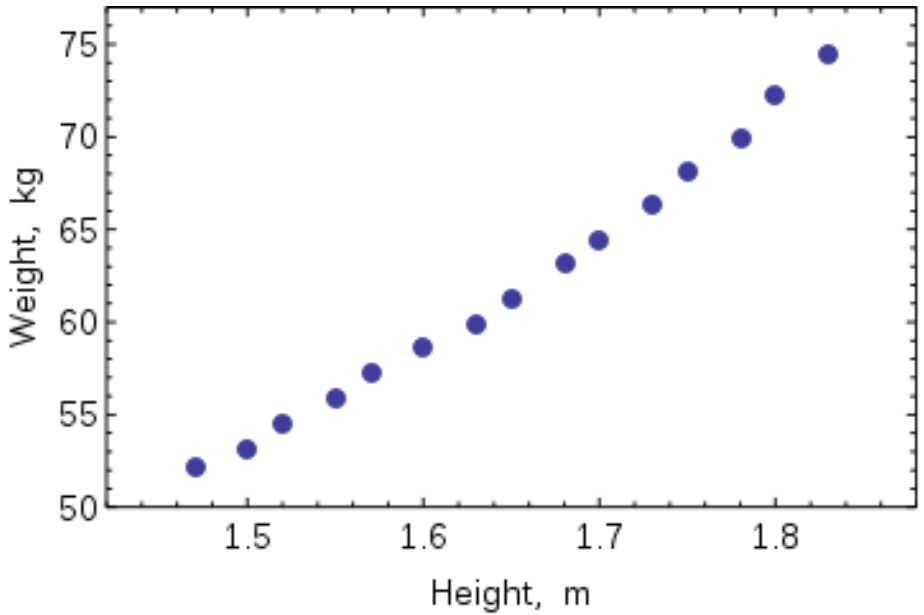
Build a line chart, with gdp_cap and life_exp.

ANSWER

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
#make plot gdp_cap on x-axis and  life_exp on y-axis
plt.plot(gdp_cap, life_exp )
#display the plot
plt.show()
```

SCATTER PLOT

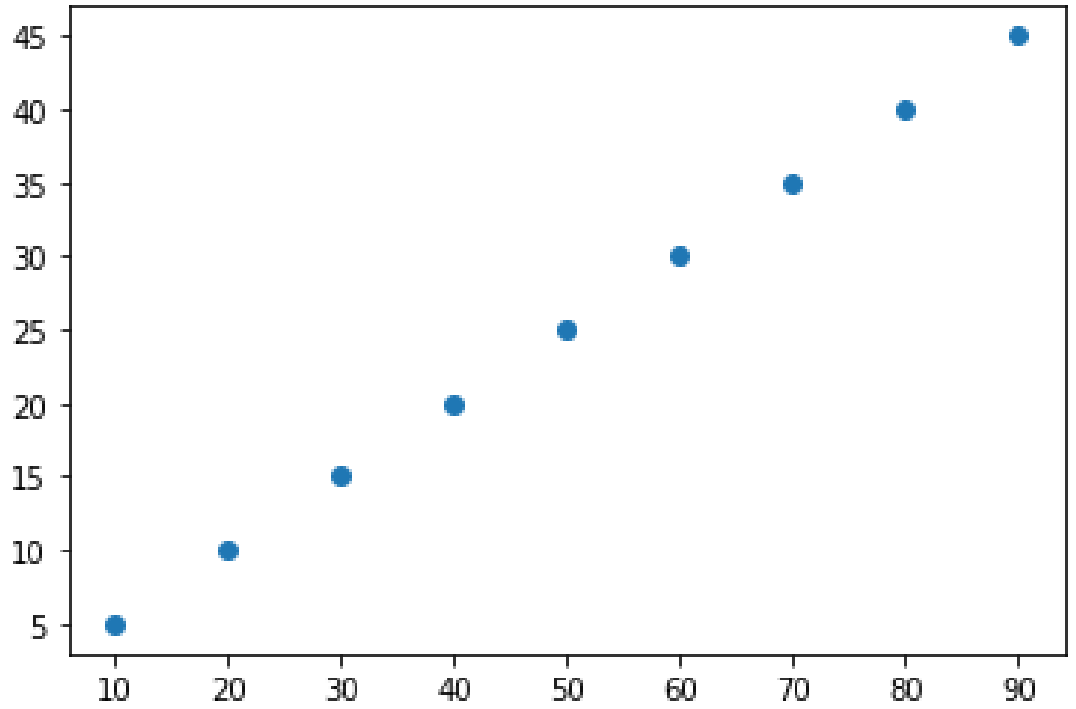
- When you have a time scale along the horizontal axis, the line plot is your friend.
- A scatter plot is a type of plot that shows the data as a collection of points. The position of a point depends on its two-dimensional value, where each value is a position on either the horizontal or vertical dimension.
- But in many other cases, when you're trying to assess if there's a correlation between two variables, the scatter plot is the better choice.
- To create a Scatter Plot use plt.scatter(x,y) where x and y represent the corresponding axis.



SCATTER PLOT -EXAMPLE

Example of how to build a scatter plot

```
#import matplotlib
import matplotlib.pyplot as plt
#adding lists
x=[10,20,30,40,50,60,70,80,90]
y=[5,10,15,20,25,30,35,40,45]
#create scatterplot
plt.scatter(x,y)
plt.show()
```



TASK

Build a scatter plot, with gdp_cap and life_exp.

- To create a Scatter Plot use plt.scatter(x,y) where x and y represent the corresponding axis.

TASK

Using the savings data create a scatter plot?

Month	Savings $
Jan	209.00
Feb	250.00
Mar	209.00
Apr	297.00
May	356.00
Jun	388.00
Jul	305.00
Aug	397.00
Sep	362.00
Oct	237.00
Nov	282.00
Dec	238.00

ANSWER

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
month=['Jen','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nev','Dec']
saving=[209.00,250.00,209.00,297.00,356.00,388.00,305.00,397.00,362.00,237.00,282.00,238.00]
#make line plot  year on x-axis and pop on y-axis
plt.scatter(month,saving)
#display the plot
plt.show()
```

MATPLOTLIB - LOGARITHM FUNCTION

A logarithm is a way to make a large number appear small by looking at it as a power of 10.

The logarithmic scale in Matplotlib

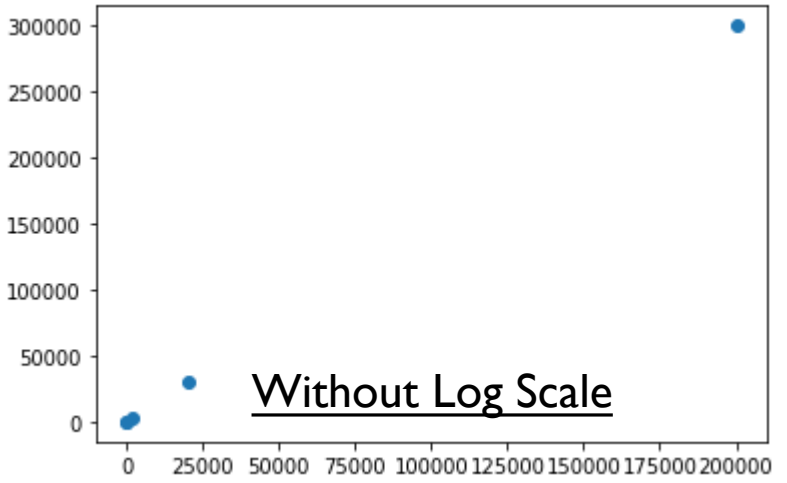
A two-dimensional chart in Matplotlib has a yscale and xscale. The scale means the graduations or tick marks along an axis. They can be any of:

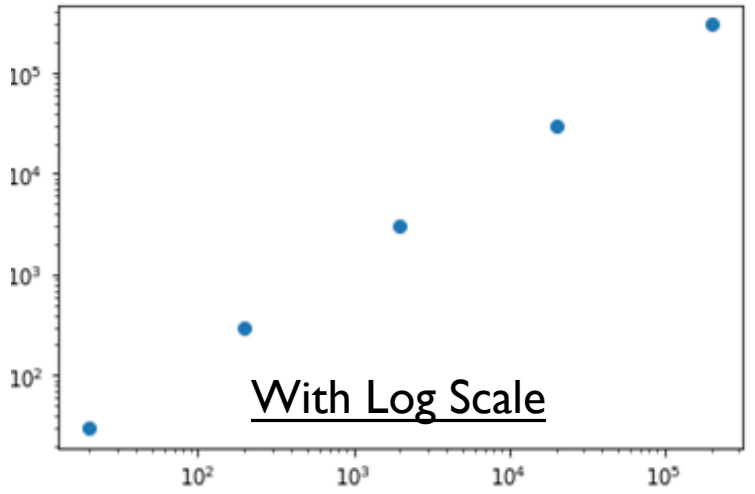
- matplotlib.scale.LinearScale—These are just numbers, like 1, 2, 3.
- matplotlib.scale.LogScale—These are powers of 10. You could use any base, like 2 or the natural logarithm value, which is given by the number e. Using different bases would narrow or widen the spacing of the plotted elements, making visibility easier.

LOGARITHM FUNCTION - EXAMPLE

Using the Log Scale

```
#importing matplotlib
import matplotlib.pyplot as plt
#creating lists
x = [20, 200, 2000, 20000, 200000]
y = [30, 300, 3000, 30000, 300000]
#creating scatter plot
plt.scatter(x, y)
#applying log on x & y axis
plt.xscale('log')
plt.yscale('log')
plt.show()
```





SCATTER PLOT – TASK BACKGROUND

Let's continue with the gdp_cap versus life_exp plot, the GDP and life expectancy data for different countries. Maybe a scatter plot will be a better alternative?

TASK

Change scatter plot by using log scale: A correlation will become clear when you display the GDP per capita on a logarithmic scale. Add the line plt.xscale('log').

Finish off your script with plt.show() to display the plot.

ANSWER

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
#make scatter plot gdp_cap on x-axis and  life_exp on y-axis
plt. scatter(gdp_cap, life_exp )
plt.xscale('log')
#display the plot
plt.show()
```

SCATTER PLOT – TASK BACKGROUND

In the previous task, you saw that that the higher GDP usually corresponds to a higher life expectancy. In other words, there is a positive correlation.

Do you think there's a relationship between population and life expectancy of a country? The list life_exp from the previous task is already available. In addition, pop is also available, listing the corresponding populations for the countries.

TASK

Build a scatter plot, where pop is mapped on the horizontal axis, and life_exp is mapped on the vertical axis. The population data is in millions of people. Convert the Population to a logarithmic Scale

Finish the script with plt.show() to actually display the plot. Do you see a correlation?

For completing this task use code below.

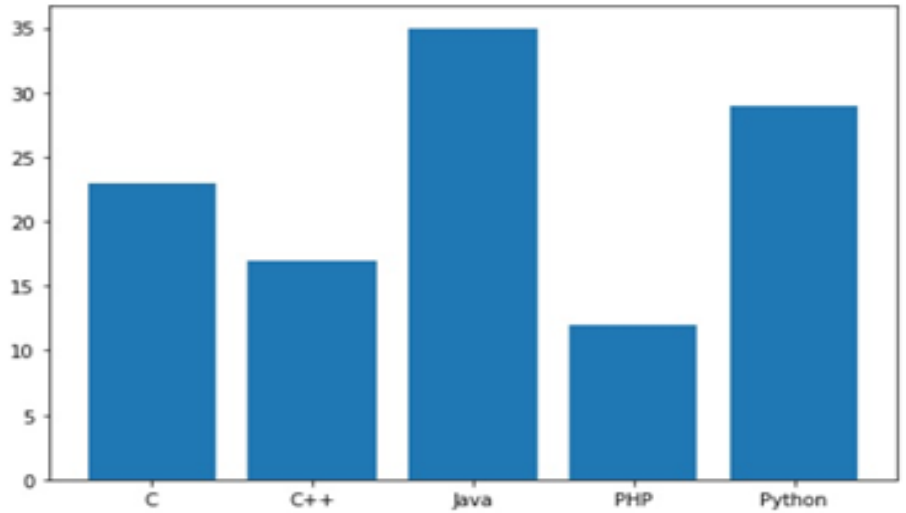
```
#import pandas
import pandas as pd
#importing population & life expectancy data using the read function from Panda's
country_Data = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/gapminder.csv')
import numpy as np
population=np.array(country_Data)
pop=(population[:,3])
life_exp =(population[:,5])
```

ANSWER

```
#import pandas
import pandas as pd
#Import numpy
import numpy as np
#importing population & life expectancy data using the read function from Panda's
country_Data = pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/gapminder.csv')
np_country_Data=np.array(country_Data)
pop=(np_country_Data[:,3])
life_exp =(np_country_Data[:,5])
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
#make scatter plot pop on x-axis and  life_exp on y-axis
plt.scatter(pop ,life_exp )
#display the plot
plt.show()
```

BUILD A HISTOGRAM

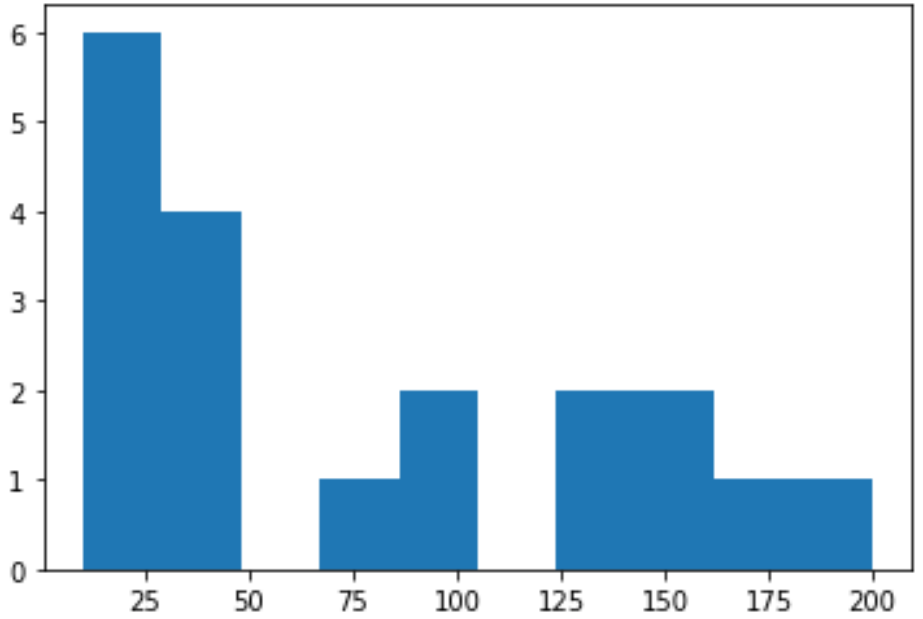
- A histogram is an accurate representation of the distribution of numerical data.
- A histogram shows the frequency on the vertical axis and the horizontal axis is another dimension.
- Usually it has bins, where every bin has a minimum and maximum value. Each bin also has a frequency between x and infinite
- To create a histogram use plt.hist(x).



HISTOGRAM - EXAMPLE

Creating Histogram

```
#importing matplotlib
import matplotlib.pyplot as plt
#creating lists
x = [10,15,20,25,40,45,70,90,100,125,125,150,150,175,20,15,40,40, 200]
#creating histogram
plt. hist(x)
plt.show()
```



TASK

Use plt.hist() to create a histogram of the values in life_exp and display the histogram.

ANSWER

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
#make histogram of  life_exp data
plt. hist(life_exp )
#display the  histogram plot
plt.show()
```

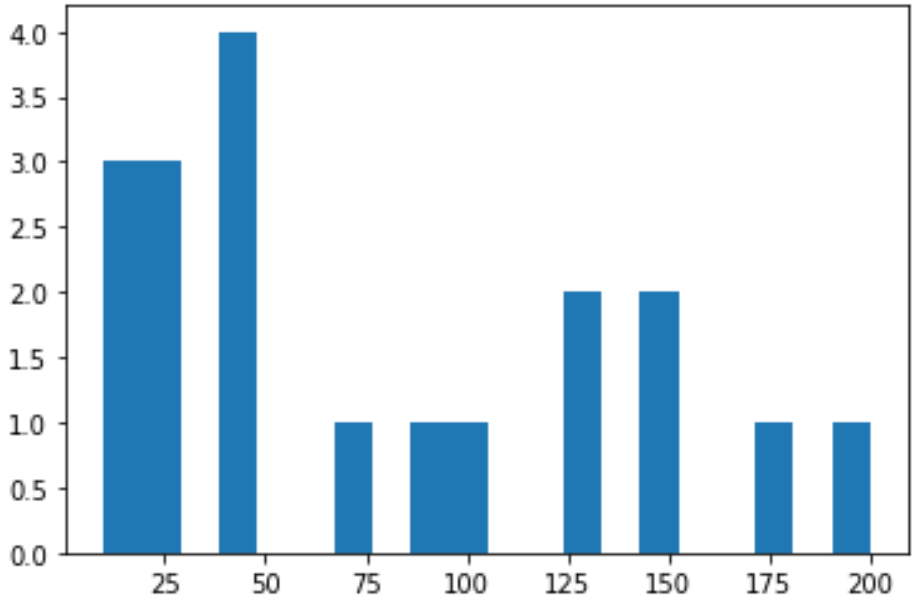
BINS IN HISTOGRAM

- Bins are the number of intervals you want to divide all of your data into, such that it can be displayed as bars on a histogram.
- In the previous task, you didn't specify the number of bins. By default, Python sets the number of bins to 10 in that case.
- The number of bins is pretty important. Too few bins will oversimplify reality and won't show you the details. Too many bins will overcomplicate reality and won't show the bigger picture.
- To control the number of bins to divide your data in, you can set the bins argument.

BINS IN HISTOGRAM - EXAMPLE

Creating Histrogram with bins

```
#importing matplotlib
import matplotlib.pyplot as plt
#creating lists
x = [10,15,20,25,40,45,70,90,100,125,125,150,150,175,20,15,40,40, 200]
#creating histogram
bin=15
plt. hist(x,bin)
plt.show()
```



BUILD A HISTOGRAM – TASK BACKGROUND

That's exactly what you'll do in this exercise. You'll be making two plots here. The code in the script already includes plt.show() and plt.clf() calls; plt.show() displays a plot; plt.clf() cleans it up again so you can start afresh.

As before, life_exp is available and matplotlib.pyplot is imported as plt.

TASK

Build a histogram of life_exp, with 5 bins. Can you tell which bin contains the most observations?

ANSWER

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
#make histogram of  life_exp data with 5 bins
plt. hist(life_exp,5)
#display the  histogram plot
plt.show()
plt.clf()
```

TASK

Build another histogram of life_exp, this time with 20 bins. Is this better?

ANSWER

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
#make histogram of  life_exp data with 20 bins
plt. hist(life_exp, 20)
#display the  histogram plot
plt.show()
plt.clf()
```

TASK

Build a histogram of life_exp with 5 Custom bins.

40 - 50
50 - 60
60 - 65
65 - 68
68 - 80

ANSWER

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
#make histogram of  life_exp data with 15 bins
plt. hist(life_exp, 15)
#display the  histogram plot
plt.show()
plt.clf()
```

TASK

Build a histogram of life_exp1950, with 15 bins.

For completing this task use code below.

life_exp1950 =
[28.8,55.23,43.08,30.02,62.48,69.12,66.8,50.94,37.48,68.0,38.22,40.41,53.82,47.62,50.92,59.6,31.98,39.03,39.42,38.52,
68.75,35.46,38.09,54.74,44.0,50.64,40.72,39.14,42.11,57.21,40.48,61.21,59.42,66.87,70.78,34.81,45.93,48.36,41.89,45.
26,34.48,35.93,34.08,66.55,67.41,37.0,30.0,67.5,43.15,65.86,42.02,33.61,32.5,37.58,41.91,60.96,64.03,72.49,37.37,37.4
7,44.87,45.32,66.91,65.39,65.94,58.53,63.03,43.16,42.27,50.06,47.45,55.56,55.93,42.14,38.48,42.72,36.68,36.26,48.46,
33.68,40.54,50.99,50.79,42.24,59.16,42.87,31.29,36.32,41.72,36.16,72.13,69.39,42.31,37.44,36.32,72.67,37.58,43.44,55
.19,62.65,43.9,47.75,61.31,59.82,64.28,52.72,61.05,40.0,46.47,39.88,37.28,58.0,30.33,60.4,64.36,65.57,32.98,45.01,64.
94,57.59,38.64,41.41,71.86,69.62,45.88,58.5,41.22,50.85,38.6,59.1,44.6,43.58,39.98,69.18,68.44,66.07,55.09,40.41,43.1
6,32.55,42.04,48.45]

ANSWER

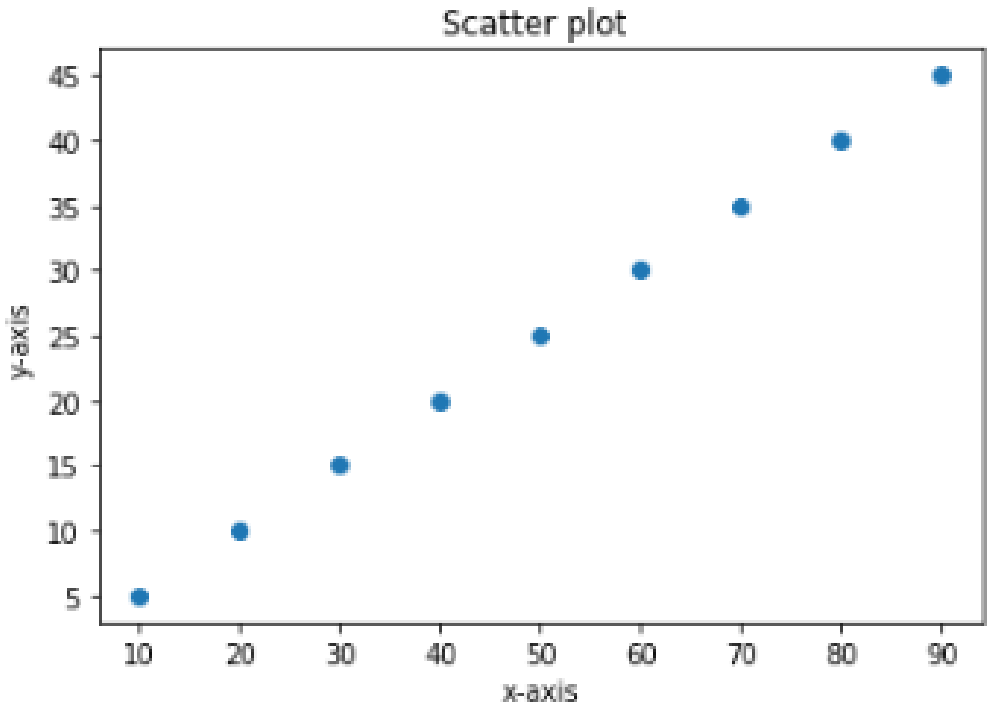
```
life_exp1950 =
[28.8,55.23,43.08,30.02,62.48,69.12,66.8,50.94,37.48,68.0,38.22,40.41,53.82,47.62,50.92,59.6,31.98,39.03,39.42,38.52,
68.75,35.46,38.09,54.74,44.0,50.64,40.72,39.14,42.11,57.21,40.48,61.21,59.42,66.87,70.78,34.81,45.93,48.36,41.89,45.
26,34.48,35.93,34.08,66.55,67.41,37.0,30.0,67.5,43.15,65.86,42.02,33.61,32.5,37.58,41.91,60.96,64.03,72.49,37.37,37.4
7,44.87,45.32,66.91,65.39,65.94,58.53,63.03,43.16,42.27,50.06,47.45,55.56,55.93,42.14,38.48,42.72,36.68,36.26,48.46,
33.68,40.54,50.99,50.79,42.24,59.16,42.87,31.29,36.32,41.72,36.16,72.13,69.39,42.31,37.44,36.32,72.67,37.58,43.44,55
.19,62.65,43.9,47.75,61.31,59.82,64.28,52.72,61.05,40.0,46.47,39.88,37.28,58.0,30.33,60.4,64.36,65.57,32.98,45.01,64.
94,57.59,38.64,41.41,71.86,69.62,45.88,58.5,41.22,50.85,38.6,59.1,44.6,43.58,39.98,69.18,68.44,66.07,55.09,40.41,43.1
6,32.55,42.04,48.45]

# Import matplotlib.pyplot as plt

import matplotlib.pyplot as plt

#make histogram of  life_exp1950 data with 15bins

plt. hist(life_exp1950, 15)

#display the  histogram plot

plt.show()
```

ADD LABELS ON PLOT

- You can use labels on your plots by adding axis labels and a title which enhance the understandability of the plot.
- You can do this with the xlabel(), ylabel() and title() functions, available in matplotlib.pyplot.
- With plt.xlabel() and plt.ylabel(), we can assign labels to those respective axis.
- You can assign the plot's title with plt.title()

ADD LABELS ON PLOT -EXAMPLE

```
#import matplotlib
import matplotlib.pyplot as plt
# Create data
x=[10,20,30,40,50,60,70,80,90]
y=[5,10,15,20,25,30,35,40,45]
# Plot
plt.scatter(x, y)
plt.title('Scatter plot ')
plt.xlabel('x-axis')
plt.ylabel('y-axi')
plt.show()
```



TASK

Using the savings data create a scatter plot? And give a title of your plot also name x & y axis months & savings respectivly.

Month	Savings $
Jan	209.00
Feb	250.00
Mar	209.00
Apr	297.00
May	356.00
Jun	388.00
Jul	305.00
Aug	397.00
Sep	362.00
Oct	237.00
Nov	282.00
Dec	238.00

ANSWER

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
month=['Jen','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nev','Dec']
saving=[209.00,250.00,209.00,297.00,356.00,388.00,305.00,397.00,362.00,237.00,282.00,238.00]
#make line plot  year on x-axis and pop on y-axis
plt.scatter(month,saving)
# Add title
plt.title('Monthly savings')
# Add axis labels
plt.xlabel('month')
plt.ylabel('saving')
#display the plot
plt.show()
```

TASK

You're going to work on the scatter plot with world development data: GDP per capita on the x-axis (logarithmic scale), life expectancy on the y-axis.

Add the customizations given below, finish the script with plt.show() to actually display the plot.

#Strings

X Axis as 'GDP per Capita [in USD]'

Y Axis as ‘Life Expectancy [in years]'

Tile of the Graph Should be 'World Development'

ANSWER

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
# Strings
xlab = 'GDP per Capita [in USD]'
ylab = 'Life Expectancy [in years]'
title = 'World Development'
# Basic scatter plot, log scale
plt.scatter(gdp_cap, life_exp)
plt.xscale('log')
# Add axis labels
plt.xlabel(xlab)
plt.ylabel(ylab)
# Add title
plt.title(title)
# After customizing, display the
plt.show()
```

TICKS

Ticks are the values used to show specific points on the coordinate axis.

It can be a number or a string.

Whenever we plot a graph, the axis adjust and take the default ticks.

Matplotlib's default ticks are generally sufficient in common situations but are in no way optimal for every plot.

Here, we will see how to customize these ticks as per our need.

plt.xticks() use to customize x-axis ticks.

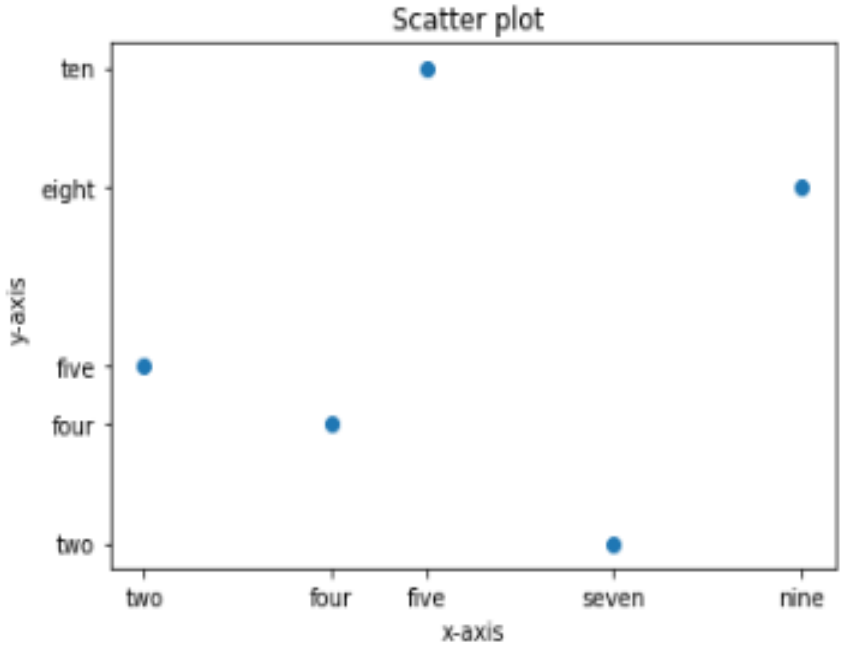
plt.yticks() use to customize y -axis ticks.

You can control xticks & yticks by specifying two arguments:

```
plt.xticks([0,1,2], ["one","two","three"])
plt.yticks([0,1,2], ["one","two","three"])
```

TICKS -EXAMPLE

```
#Create scatter plot of using X & Y values
# importing matplotlib module
import matplotlib.pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7]
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot
plt.scatter(x, y)
plt.title('Scatter plot ')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
# Adapt the ticks on the x-axis & y-axis
plt.xticks([5, 2, 9, 4, 7], ["five","two","nine","four","seven"])
plt.yticks([10, 5, 8, 4, 2], ["ten","five","eight","four","two"])
plt.show()
```



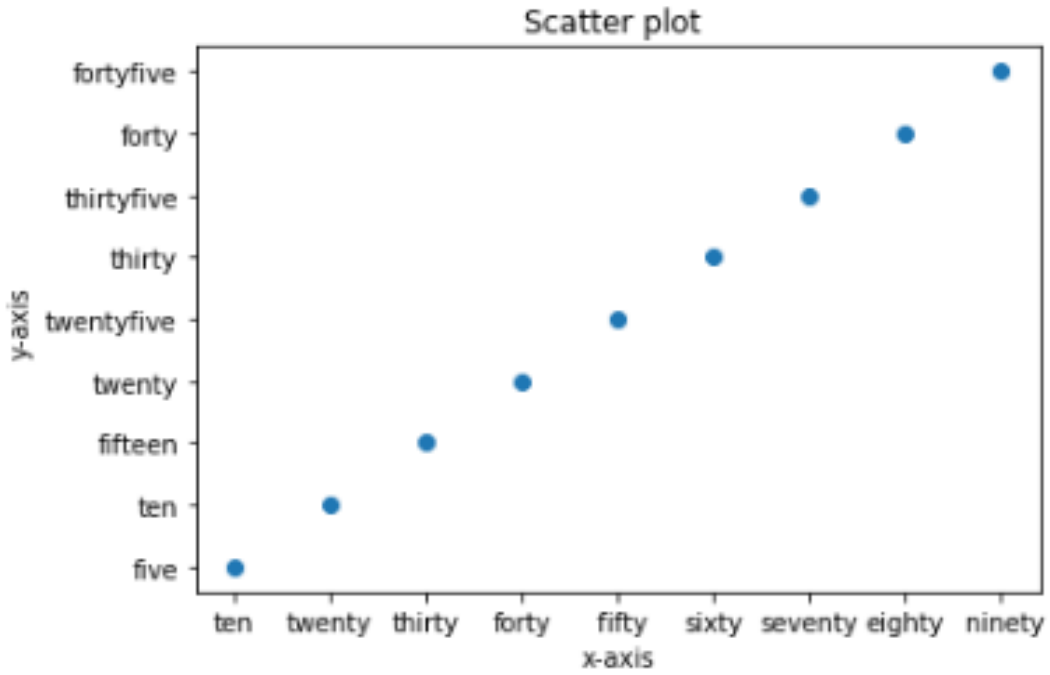
TASK

Create a scatter plot using x & y values show strings instead of numbers

x=[10,20,30,40,50,60,70,80,90]

y=[5,10,15,20,25,30,35,40,45]

Your answer should look like:



ANSWER

```
#Create scatter plot of using X & Y values
# importing matplotlib module
import matplotlib.pyplot as plt
x=[10,20,30,40,50,60,70,80,90]
y=[5,10,15,20,25,30,35,40,45]
# Function to plot
plt.scatter(x, y)
plt.title('Scatter plot ')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
# Adapt the ticks on the x-axis & y-axis
plt.xticks([10,20,30,40,50,60,70,80,90], ['ten', 'twenty', 'thirty', 'forty',' fifty' , 'sixty' ,'seventy', 'eighty',' ninety' ] )
plt.yticks([5,10,15,20,25,30,35,40,45], [ 'five', 'ten', 'fifteen', 'twenty', 'twentyfive'  ,'thirty' ,'thirtyfive'  ,'forty'  ,'fortyfive'])
plt.show()
```

TICKS – TASK BACKGROUND

Let's do a similar thing for the x-axis of your world development chart, with the xticks() function.

The tick values 1000, 10000 and 100000 should be replaced by 1k, 10k and 100k.

Prepare the two lists: tick_val and tick_lab.

TASK

Let's do a similar thing for the x-axis of your world development chart(GDP, Life Expectancy), with the xticks() function.

The tick values 1000, 10000 and 100000 should be replaced by 1k, 10k and 100k. (GDP)

Prepare the two lists: tick_val and tick_lab.
Use tick_val and tick_lab as inputs to the xticks() function to make the the plot more readable.

As usual, display the plot with plt.show() after you've added the customizations.

```
tick_val = [1000, 10000, 100000]
tick_lab = ['1k', '10k', '100k']
```

ANSWER

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
# Strings
xlab = 'GDP per Capita [in USD]'
ylab = 'Life Expectancy [in years]'
title = 'World Development '
# Definition of tick_val and tick_lab
tick_val = [1000, 10000, 100000]
tick_lab = ['1k', '10k', '100k']
# Basic scatter plot, log scale
plt.scatter(gdp_cap, life_exp)
plt.xscale('log')
# Add axis labels
plt.xlabel(xlab)
plt.ylabel(ylab)
# Add title
plt.title(title)
# Adapt the ticks on the x-axis
plt.xticks(tick_val,tick_lab)
# After customizing, display the
plt.show()
```

SIZES

Right now, the scatter plot is just a cloud of blue dots, indistinguishable from each other. Let's change this. Wouldn't it be nice if the size of the dots corresponds to the population?

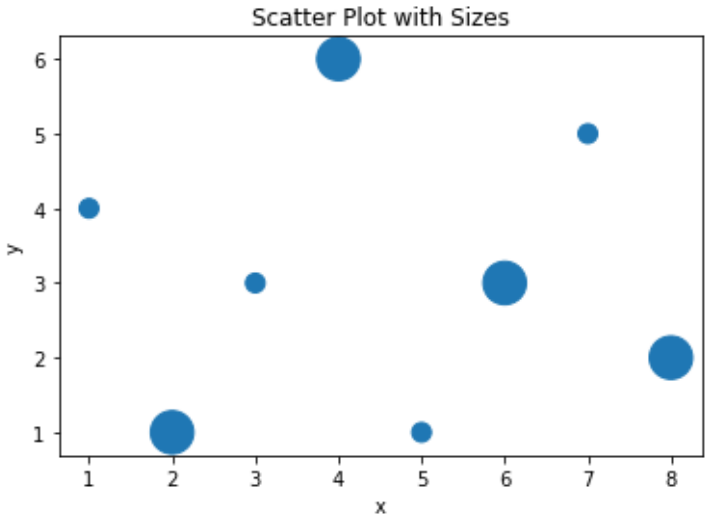
To accomplish use the pop list your created. It contains population numbers for each country expressed in millions. You can see that this list is added to the scatter method, as the argument plt.scatter(x,y,s=size)

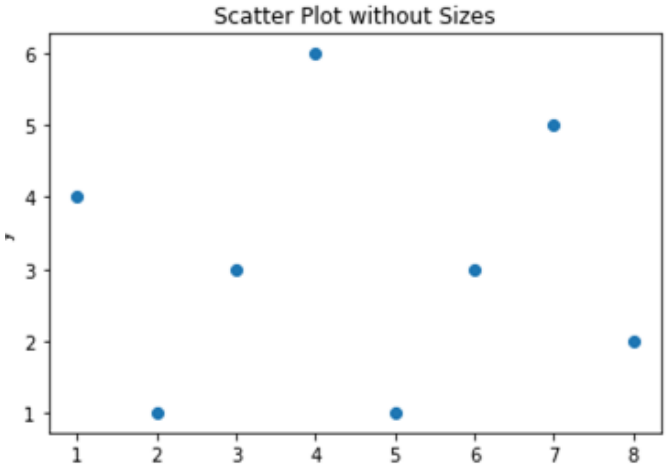
, for size. Run the script to see how the plot changes.

Looks good, but increasing the size of the bubbles will make things stand out more.

SIZES-EXAMPLE

```
#import matplotlib
import matplotlib.pyplot as plt
#adding variables
x = [1,2,3,4,5,6,7,8]
y = [4,1,3,6,1,3,5,2]
size = [100,500,100,500,100,500,100,500]
#creating scatter plot
plt.scatter(x,y,s=size)
plt.title('Scatter plot with Sizes')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```





TASK

Use Pandas read function, bring the population data from github using the following link and convert it into a numpy array named np_pop.

https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/popdata1.csv

ANSWER

```
# Import numpy as np

import numpy as np

# Store pop as a numpy array: np_pop

population_Data
=pd.read_csv('https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/popdata1.csv')

pop=np.array(population_Data)

np_pop=(pop[:,0])
```

TASK

Create a Scatter Plot Using the GDP, Life Expectance and Population

Change the s argument inside plt.scatter() to be equal to np_pop (Population).

Use below code for customizations of graph

```
plt.xscale('log')
plt.xlabel('GDP per Capita [in USD]')
plt.ylabel('Life Expectancy [in years]')
plt.title('World Development in 2007')
```

ANSWER

```
# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
# Import numpy as np
import numpy as np
# Store pop as a numpy array: np_pop
np_pop=np.array(pop)
# Double np_pop
np_pop=np_pop*2
# Update: set s argument to np_pop
plt.scatter(gdp_cap, life_exp, s =np_pop)
# Previous customizations
plt.xscale('log')
plt.xlabel('GDP per Capita [in USD]')
plt.ylabel('Life Expectancy [in years]')
plt.title('World Development in 2007')
plt.xticks([1000, 10000, 100000],['1k', '10k', '100k'])
# Display the plot
plt.show()
```

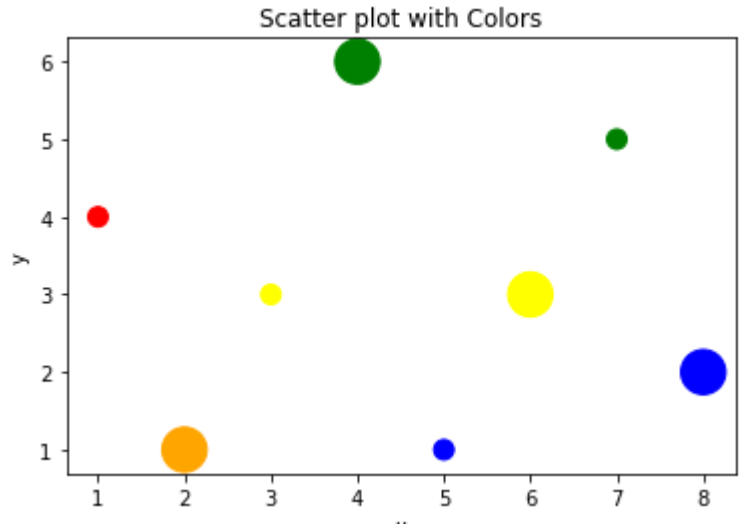
ADDING COLORS

You can add colors on your scatter plot by using the following code: plt.scatter(x,y,s=size,c=plotcolor)

This will give your presentation much more visual effect and will enhance understandablity.

Using Colors in a Scatter plot

```
#importing matplotlib
import matplotlib.pyplot as plt
#creating the x & y axis
x = [1,2,3,4,5,6,7,8]
y = [4,1,3,6,1,3,5,2]
size = [100,500,100,500,100,500,100,500]
plotcolor = ['red','orange','yellow','green','blue','yellow','green','blue']
#adding colors
plt.scatter(x,y,s=size,c=plotcolor)
#creating scatter plot
plt.title('Scatter plot with Colors')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



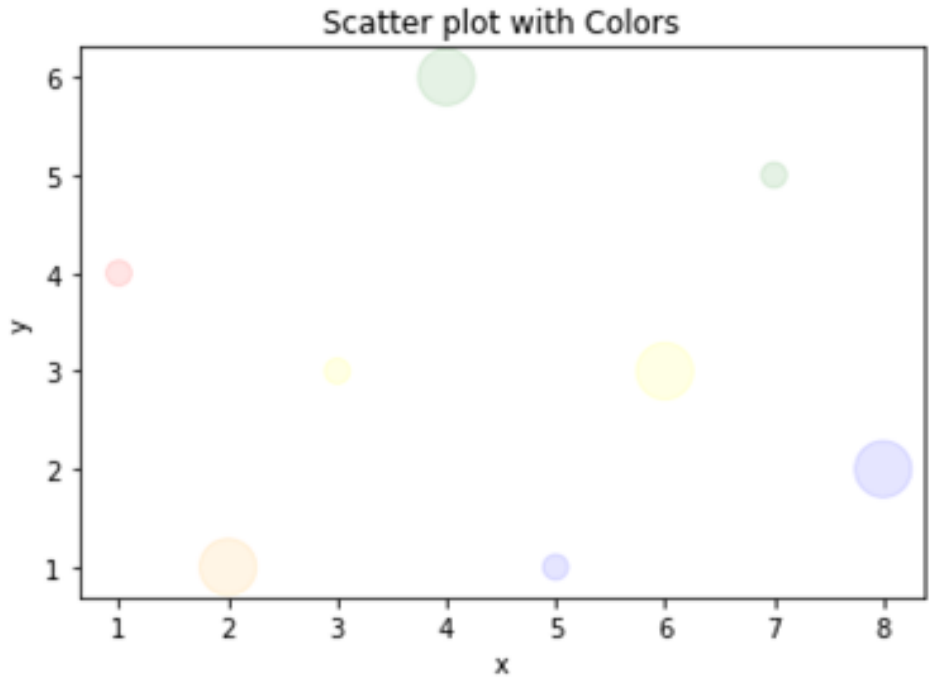
CHANGING OPACITY

- So when you create a plot, by default, matplotlib will have the default transparency set (a transparency of 1).
- Matplotlib allows you to adjust the transparency of a graph plot using the alpha attribute.
- By default, alpha=1
- If you want to make the graph plot more transparent, then you can make alpha less than 1, such as 0.5 or 0.25.
- If you want to make the graph plot less transparent, then you can make alpha greater than 1.
- You can use the following code to change the opacity of your grid: plt.grid(x-position, y-position, alpha = 0.5)

CHANGING OPACITY

Using opacity function in matplotlib

```
#import matplotlib
import matplotlib.pyplot as plt
#creating x & y values
x = [1,2,3,4,5,6,7,8]
y = [4,1,3,6,1,3,5,2]
size = [100,500,100,500,100,500,100,500]
plotcolor = ['red','orange','yellow','green','blue','yellow','green','blue']
#creating a scatter plot and decreasing opacity
plt.scatter(x,y,s=size,c=plotcolor ,alpha=0.1)
#adding titles & lables
plt.title('Scatter plot with Colors')
plt.xlabel('x')
plt.ylabel('y')
#show plot
plt.show()
```



TASK

Using the last plot now add color to your plot. Change the opacity of the bubbles by setting the alpha argument to 0.8 inside plt.scatter(). Alpha can be set from zero to one, where zero is totally transparent, and one is not at all transparent.

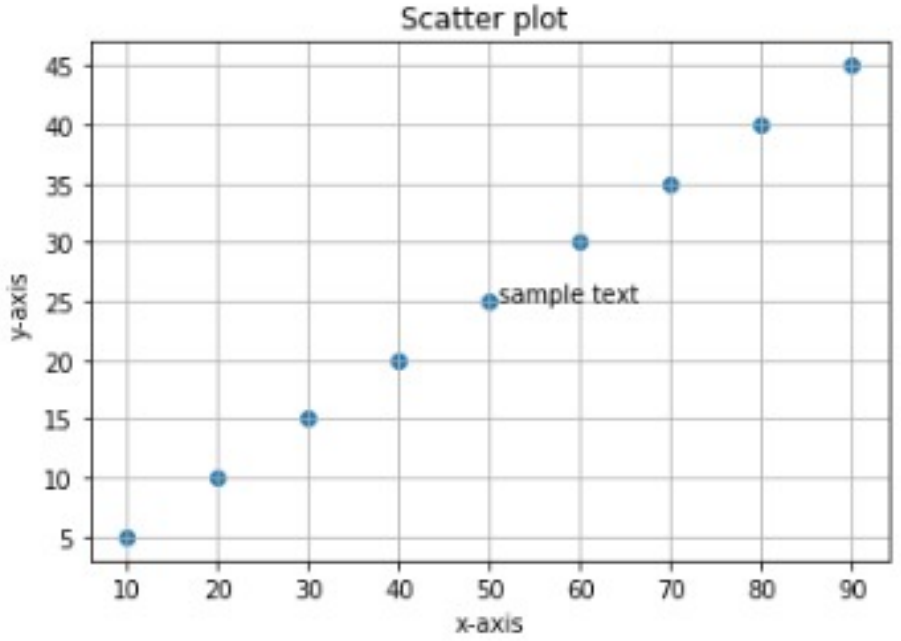
```
col = ['red', 'green', 'blue', 'blue', 'yellow', 'black', 'green', 'red', 'red', 'green', 'blue', 'yellow', 'green', 'blue', 'yellow', 'green', 'blue', 'blue', 'red', 'blue', 'yellow', 'blue', 'blue', 'yellow', 'red', 'yellow', 'blue', 'blue', 'blue', 'yellow', 'blue', 'green', 'yellow', 'green', 'green', 'blue', 'yellow', 'yellow', 'blue', 'yellow', 'blue', 'blue', 'blue', 'green', 'green', 'blue', 'blue', 'green', 'blue', 'green', 'yellow', 'blue', 'blue', 'yellow', 'yellow', 'red', 'green', 'green', 'red', 'red', 'red', 'red', 'green', 'red', 'green', 'yellow', 'red', 'red', 'blue', 'red', 'red', 'red', 'red', 'blue', 'blue', 'blue', 'blue', 'blue', 'red', 'blue', 'blue', 'blue', 'yellow', 'red', 'green', 'blue', 'blue', 'red', 'blue', 'red', 'green', 'black', 'yellow', 'blue', 'blue', 'green', 'red', 'red', 'yellow', 'yellow', 'yellow', 'red', 'green', 'green', 'yellow', 'blue', 'green', 'blue', 'blue', 'red', 'blue', 'green', 'blue', 'red', 'green', 'green', 'blue', 'blue', 'green', 'red', 'blue', 'blue', 'green', 'green', 'red', 'red', 'blue', 'red', 'blue', 'yellow', 'blue', 'green', 'blue', 'green', 'yellow', 'yellow', 'yellow', 'red', 'red', 'red', 'blue', 'blue']
```

ANSWER

```
import matplotlib.pyplot as plt
import numpy as np
col = ['red', 'green', 'blue', 'blue', 'yellow', 'black', 'green', 'red', 'red', 'green', 'blue', 'yellow', 'green', 'blue', 'yellow', 'green', 'blue', 'blue', 'red', 'blue', 'yellow', 'blue', 'blue', 'yellow',
'red', 'yellow', 'blue', 'blue', 'blue', 'yellow', 'blue', 'green', 'yellow', 'green', 'green', 'blue', 'yellow', 'yellow', 'blue', 'yellow', 'blue', 'blue', 'blue', 'green', 'green', 'blue', 'blue', 'green',
'blue', 'green', 'yellow', 'blue', 'blue', 'yellow', 'yellow', 'red', 'green', 'green', 'red', 'red', 'red', 'red', 'green', 'red', 'green', 'yellow', 'red', 'red', 'blue', 'red', 'red', 'red', 'red', 'blue',
'blue', 'blue', 'blue', 'blue', 'red', 'blue', 'blue', 'blue', 'yellow', 'red', 'green', 'blue', 'blue', 'red', 'blue', 'red', 'green', 'black', 'yellow', 'blue', 'blue', 'green', 'red', 'red', 'yellow', 'yellow',
'yellow', 'red', 'green', 'green', 'yellow', 'blue', 'green', 'blue', 'blue', 'red', 'blue', 'green', 'blue', 'red', 'green', 'green', 'blue', 'blue', 'green', 'red', 'blue', 'blue', 'green', 'green', 'red',
'red', 'blue', 'red', 'blue', 'yellow', 'blue', 'green', 'blue', 'green', 'yellow', 'yellow', 'yellow', 'red', 'red', 'red', 'blue', 'blue']
# Specify c and alpha inside plt.scatter()
plt.scatter(x = gdp_cap, y = life_exp, s = np.array(pop) * 2,c=col,alpha=0.8)
# Previous customizations
plt.xscale('log')
plt.xlabel('GDP per Capita [in USD]')
plt.ylabel('Life Expectancy [in years]')
plt.title('World Development in 2007')
plt.xticks([1000,10000,100000], ['1k','10k','100k'])
# Show the plot
plt.show()
```

GRID & TEXT ON PLOT

- You can use a grid & text on your plot to make the visualization more understandable.
- You can use plt.grid(True) command to add a grid to your plot.
- You can use plt.text(x-position, y-position, 'sample text') command to add text to your plot.



TASK

Add two plt.text() function. They should add the words "India" and "China" in the plot.

```
# Additional customizations
plt.text(5300, 80, 'India')
plt.text(58700, 80, 'China')
```

Add plt.grid(True) after the plt.text() calls so that gridlines are drawn on the plot.

ANSWER

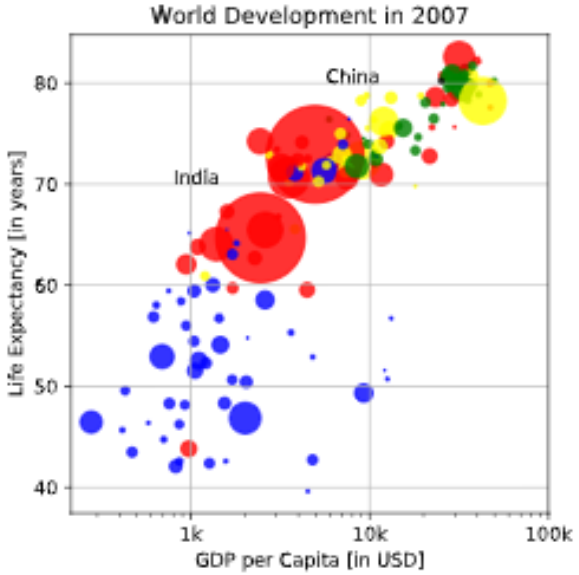
```
import matplotlib.pyplot as plt
import numpy as np
# Specify c and alpha inside plt.scatter()
plt.scatter(x = gdp_cap, y = life_exp, s = np.array(pop) * 2,c=col,alpha=0.8)
# Previous customizations
plt.xscale('log')
plt.xlabel('GDP per Capita [in USD]')
plt.ylabel('Life Expectancy [in years]')
plt.title('World Development in 2007')
plt.xticks([1000,10000,100000], ['1k','10k','100k'])
# Additional customizations
plt.text(5300, 80, 'India')
plt.text(58700, 80, 'China')
# Add grid()
plt.grid(True)
# Show the plot
plt.show()
```

INTERPRETATION

If you have a look at your colorful plot, it's clear that people live longer in countries with a higher GDP per capita.

No high income countries have really short life expectancy, and no low income countries have very long life expectancy. Still, there is a huge difference in life expectancy between countries on the same income level.

Most people live in middle income countries where difference in lifespan is huge between countries; depending on how income is distributed and how it is used.



Q&A

python

THANK YOU