KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

Deemed to be University U/S 3 of UGC Act, 1956

School of Computer Engineering KIIT deemed to be University

Intro to Python basics

(6th Semester)

What is Python

- **Python** is a general purpose, dynamic, high-level, and interpreted programming language. It supports **Object Oriented programming approach** to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

- **Python is easy to learn** yet powerful and versatile scripting language, which makes it attractive for Application Development.

- **Python supports multiple programming pattern**, including object-oriented, and functional or procedural programming styles.

What is Python cont

- Python is not intended to work in a particular area, such as web programming. That is why it is known as **multipurpose programming language** because it can be used with web, enterprise, 3D CAD, etc.

- We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to assign an integer value in an integer variable.

- **Python** makes the development and debugging fast because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

Java vs Python Program

```
public class HelloWorld
{
 public static void main(String[] args)
{
  System.out.println("Hello World");
 }
 }
```

```
print("Hello World")
```

Both programs will print the same result, but it takes only one statement without using a semicolon or curly braces in Python.

Python Basic Syntax

```
def func():
      statement 1
      statement 2
      ………………….
      ………………….
        statement N
```

Here is no use of curly braces or semicolon in Python programming language. It is English-like language.

Python History

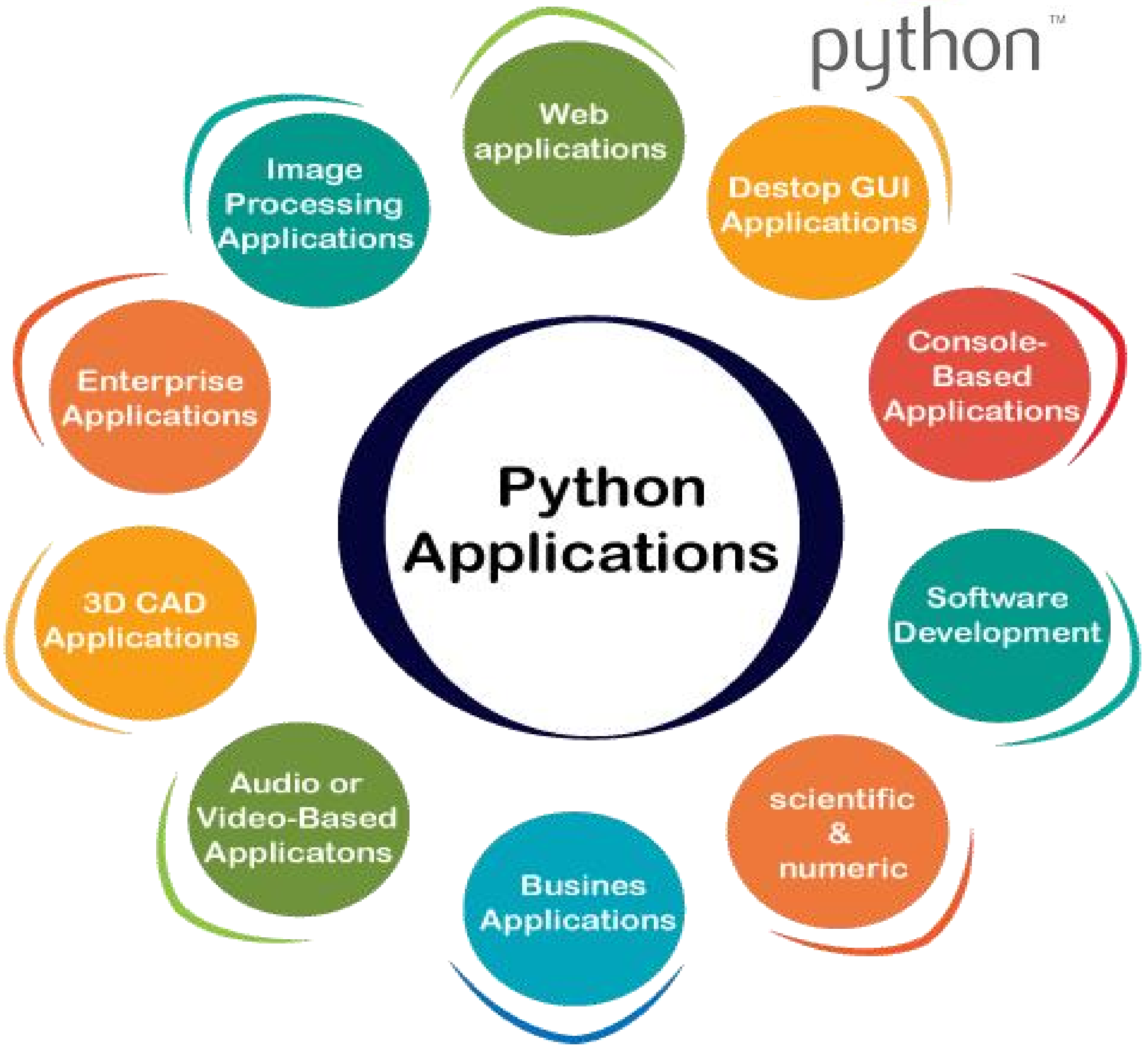
Python was invented by **Guido van Rossum in 1991 at CWI in Netherland.**

There is also a fact behind the choosing name Python. **Guido van Rossum** was a fan of the popular BBC comedy show of that time, **"Monty Python's Flying Circus"**. So he decided to pick the name Python for his newly created programming language.

Python has the vast community across the world and releases its version within the short period.

Where is Python used?

- Data Science
- Date Mining
- Desktop Applications
- Console-based Applications
- Mobile Applications
- Software Development
- Artificial Intelligence
- Web Applications
- Enterprise Applications
- 3D CAD Applications
- Machine Learning
- Computer Vision or Image Processing Applications.
- Speech Recognitions



Python Popular Frameworks and Libraries

Web development (Server-side) - Django Flask, Pyramid, CherryPy

GUIs based applications - Tk, PyGTK, PyQt, PyJs, etc.

Machine Learning - TensorFlow, PyTorch, Scikit-learn, Matplotlib, Scipy, etc.

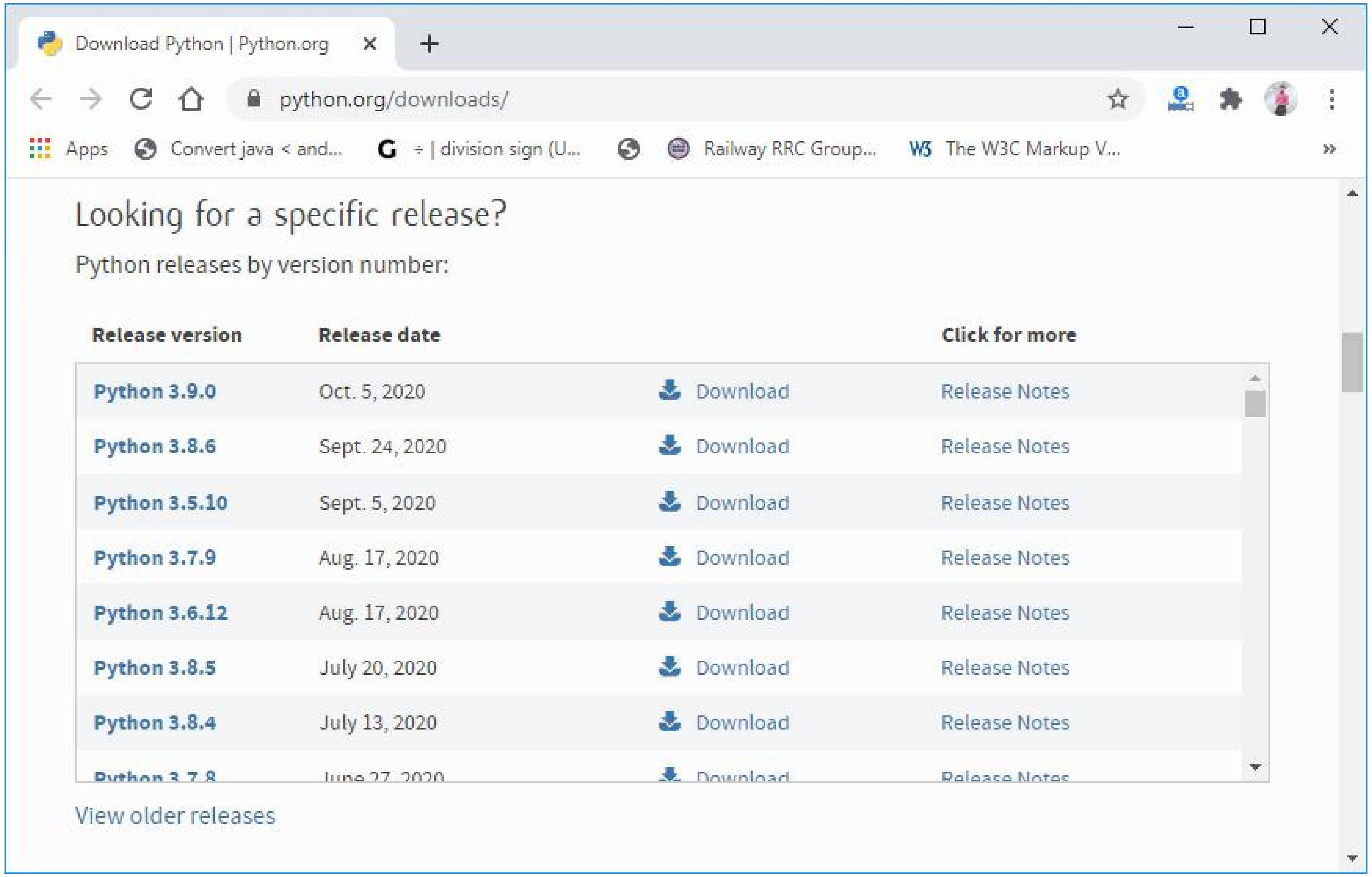
Mathematics - Numpy, Pandas, etc.

Python Version List

Python Version	Released Date
Python 1.0	Jan-94
Python 1.5	31-Dec-97
Python 1.6	05-Sep-00
Python 2.0	16-Oct-00
Python 2.1	April 17, 2001
Python 2.2	21-Dec-01
Python 2.3	29-Jul-03
Python 2.4	30-Nov-04
Python 2.5	19-Sep-06
Python 2.6	01-Oct-08
Python 2.7	03-Jul-10
Python 3.0	03-Dec-08
Python 3.1	27-Jun-09
Python 3.2	20-Feb-11
Python 3.3	29-Sep-12
Python 3.4	16-Mar-14
Python 3.5	13-Sep-15
Python 3.6	23-Dec-16
Python 3.7	27-Jun-18
Python 3.8	**14-Oct-19**

Installation on Windows

Visit the link **https://www.python.org/downloads/** to download the latest release of Python.


Download Python | Python.org

python.org/downloads/

Apps · Convert java < and... · ÷ | division sign (U... · Railway RRC Group... · The W3C Markup V...

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.9.0	Oct. 5, 2020	Download	Release Notes
Python 3.8.6	Sept. 24, 2020	Download	Release Notes
Python 3.5.10	Sept. 5, 2020	Download	Release Notes
Python 3.7.9	Aug. 17, 2020	Download	Release Notes
Python 3.6.12	Aug. 17, 2020	Download	Release Notes
Python 3.8.5	July 20, 2020	Download	Release Notes
Python 3.8.4	July 13, 2020	Download	Release Notes
Python 3.7.8	June 27, 2020	Download	Release Notes

View older releases

Taking Input to the User

Python provides the input() function which is used to take input from the user. Let's understand the following example.

Example -

```
name = input("Enter a name of student:")
print("The student name is: ", name)
```

Example -

```
a  = int(input("Enter first number: "))
b = int(input("Enter second number: "))

print(a+b)
```

Declaring Variable and Assigning Values

- Python does not bind us to declare a variable before using it in the application. It allows us to create a variable at the required time.
- We don't need to declare explicitly variable in Python. When we assign any value to the variable, that variable is declared automatically.
- The equal (=) operator is used to assign value to a variable.

Object References

- It is necessary to understand how the Python interpreter works when we declare a variable. The process of treating variables is somewhat different from many other programming languages.
- Python is the highly object-oriented programming language; that's why every data item belongs to a specific type of class.

```
print("John")
```

Output:

```
John
```

In the above print statement, we have created a string object.

```
type("John")
```

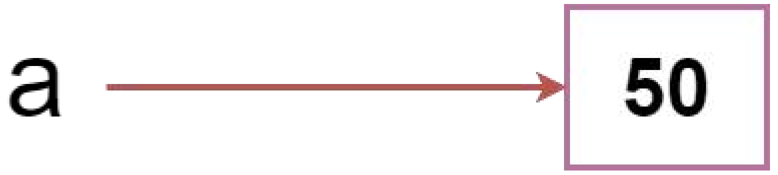
Output:

```
<class 'str'>
```

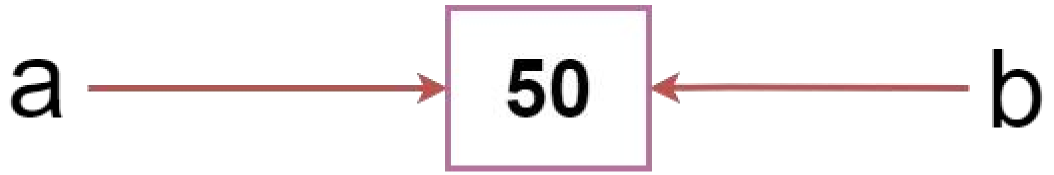
In Python, variables are a symbolic name that is a reference or pointer to an object

The variables are used to denote objects by that name.

a = 50

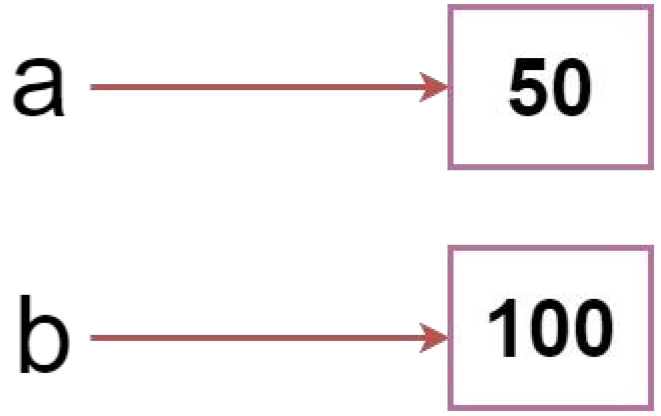


Suppose we assign the integer value 50 to a new variable b.



- The variable b refers to the same object that a points to because **Python does not create another object.**

```
a = 50

b =100
```



Python manages memory efficiently if we assign the same variable to two different values.

Object Identity

In Python, every created object identifies uniquely in Python. Python provides the guaranteed that no two objects will have the same identifier. The built-in **id() function**, is used to identify the object identifier. Consider the following example.

```
a = 50
b = a
print(id(a))
print(id(b))
# Reassigned variable a
a = 500
print(id(a))
Output:

140734982691168
140734982691168
2822056960944
```

Python Data Types

- Variables can hold values, and every value has a data-type. Python is a dynamically typed language; hence we do not need to define the type of the variable while declaring it. The interpreter implicitly binds the value with its type.

```
a = 5
```

- The variable a holds integer value five and we did not define its type. Python interpreter will automatically interpret variables a as an integer type

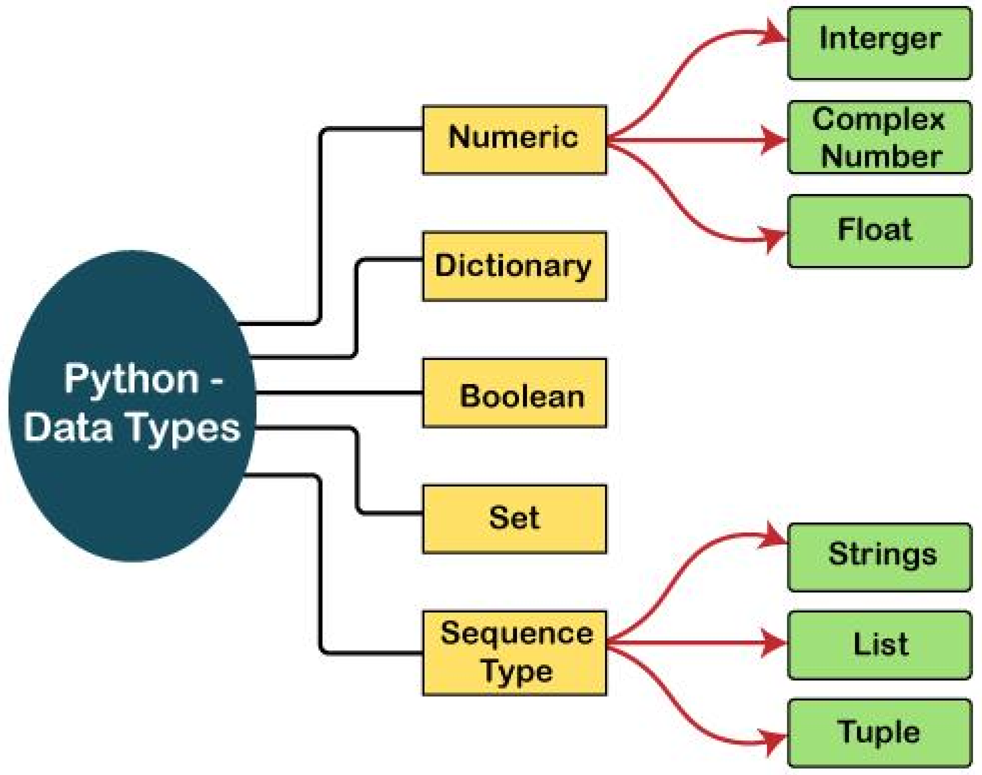
```
a=10
b="Hi Python"
c = 10.5
print(type(a))
print(type(b))
print(type(c))
```

Output:

```
<type 'int'>
<type 'str'>
<type 'float'>
```

Standard data types

- A variable can hold different types of values. For example, a person's name must be stored as a string whereas its id must be stored as an integer.
- Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.



Numbers

- Number stores numeric values. The integer, float, and complex values belong to a Python Numbers data-type.
- Python provides the type() function to know the data-type of the variable. Similarly, the isinstance() function is used to check an object belongs to a particular class.

- Python creates Number objects when a number is assigned to a variable. For example;

```
a = 5
print("The type of a", type(a))

b = 40.5
print("The type of b", type(b))

c = 1+3j
print("The type of c", type(c))
print(" c is a complex number", isinstance(1+3j,complex))
```

String

```
str1 = 'welcome to KIIT' #string str1

str2 = ' how are you' #string str2

print (str1[0:2]) #printing first two character using slice operator
print (str1[4]) #printing 4th character of the string
print (str1*2) #printing the string twice
print (str1 + str2) #printing the concatenation of str1 and str2
```

List

- Python Lists are similar to arrays in C. However, the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [].

- We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (*) works with the list in the same way as they were working with the strings.

List

```
list1  = [1, "hi", "Python", 2]
#Checking type of given list
print(type(list1))

#Printing the list1
print (list1)

# List slicing
print (list1[3:])

# List slicing
print (list1[0:2])

# List Concatenation using + operator
print (list1 + list1)

# List repetation using * operator
print (list1 * 3)
```

Tuple

- A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of the items of different data types. The items of the tuple are separated with a comma (,) and enclosed in parentheses ().

- A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple.

Dictionary

Dictionary is an unordered set of a key-value pair of items. It is like an associative array or a hash table where each key stores a specific value. Key can hold any primitive data type, whereas value is an arbitrary Python object.

The items in the dictionary are separated with the comma (,) and enclosed in the curly braces {}.

Dictionary

```
d = {1:'Jimmy', 2:'Alex', 3:'john', 4:'mike'}

# Printing dictionary
print (d)

# Accesing value using keys
print("1st name is "+d[1])
print("2nd name is "+ d[4])

print (d.keys())
print (d.values())
```

The if statement

```
num = int(input("enter the number?"))
if num%2 == 0:
    print("Number is even")
```

```
a = int(input("Enter a? "));
b = int(input("Enter b? "));
c = int(input("Enter c? "));
if a>b and a>c:
    print("a is largest");
if b>a and b>c:
    print("b is largest");
if c>a and c>b:
    print("c is largest");
```

```
marks = int(input("Enter the marks? "))
f marks > 85 and marks <= 100:
   print("Congrats ! you scored grade A ...")
lif marks > 60 and marks <= 85:
   print("You scored grade B + ...")
lif marks > 40 and marks <= 60:
   print("You scored grade B ...")
lif (marks > 30 and marks <= 40):
   print("You scored grade C ...")
lse:
   print("Sorry you are fail ?")
```

The for Loop

Python's for loop is designed to repeatedly execute a code block while iterating through a list, tuple, dictionary, or other iterable objects of Python. The process of traversing a sequence is known as iteration.

Syntax of the for Loop

```
for value in sequence:
    { code block }
```

In this case, the variable value is used to hold the value of every item present in the sequence before the iteration begins until this particular iteration is completed.

Loop iterates until the final item of the sequence are reached.

```
# Python program to show how the for loop works

# Creating a sequence which is a tuple of numbers
numbers = [4, 2, 6, 7, 3, 5, 8, 10, 6, 1, 9, 2]

# variable to store the square of the number
square = 0

# Creating an empty list
squares = []

# Creating a for loop
for value in numbers:
    square = value ** 2
    squares.append(square)
print("The list of squares is", squares)
```

```
# Python program to show how if-else statements work

string = "Python Loop"

# Initiating a loop
for s in a string:
    # giving a condition in if block
    if s == "o":
        print("If block")
    # if condition is not satisfied then else block will be executed
    else:
        print(s)
```

What is NumPy

- **NumPy stands for numeric python** which is a python package for the computation and processing of the multidimensional and single dimensional array elements.

- **Travis Oliphant** created NumPy package in 2005 by injecting the features of the ancestor module Numeric into another module Numarray.

- It is an extension module of Python which is mostly written in C. It provides various functions which are capable of performing the numeric computations with a high speed.

- **NumPy** provides various powerful data structures, implementing multi-dimensional arrays and matrices. These data structures are used for the optimal computations regarding arrays and matrices.

The need of NumPy

With the revolution of data science, data analysis libraries like NumPy, SciPy, Pandas, etc. have seen a lot of growth. With a much easier syntax than other programming languages, python is the first choice language for the data scientist.

NumPy provides a convenient and efficient way to handle the **vast amount of data**. NumPy is also very convenient with Matrix multiplication and data reshaping.

1. NumPy performs array-oriented computing.
2. It efficiently implements the multidimensional arrays.
3. It performs scientific computations.
4. It is capable of performing Fourier Transform and reshaping the data stored in multidimensional arrays.
5. NumPy provides the in-built functions for linear algebra and random number generation.

The need of NumPy

If you are going to work on **data analysis or machine learning projects**, then having a solid understanding of numpy is nearly mandatory.

Because other packages for data analysis (like pandas) is built on top of numpy and the scikit-learn package which is used to build machine learning applications works heavily with numpy as well.

So what does numpy provide?

At the core, numpy provides the excellent ndarray objects, **short for n-dimensional arrays.**

You might wonder, ‘I can store numbers and other objects in a python list itself and do all sorts of computations and manipulations through list comprehensions, for-loops etc. What do I need a numpy array for?’

- The key difference between an array and a list is, **arrays are designed to handle vectorized operations** while a python list is not.

- That means, if you apply a function it is performed on every item in the array, rather than on the whole array object.

Let’s suppose you want to **add the number 2 to every item in the list**.

```
list1 + 2  # error
```

That was not possible with a list. But you can do that on a ndarray.

```
# Add 2 to each element of arr1d

arr1d + 2

#> array([2, 3, 4, 5, 6])
```

Once a numpy array is created, you cannot increase its size. To do so, you will have to create a new array..

So many advantages..............................

NumPy Environment Setup

NumPy doesn't come bundled with Python. We have to install it using the python pip installer. Execute the following command.

```
$ pip install numpy
```

NumPy Ndarray

Ndarray is the n-dimensional array object defined in the numpy which stores the collection of the similar type of elements.

The **ndarray** object can be accessed by using the 0 based indexing. Each element of the Array object contains the same size in the memory.

Creating a ndarray object

The ndarray object can be created by using the array routine of the numpy module. For this purpose, we need to import the numpy.

```
>>> a = numpy.array
```

Creating Arrays:

- np.array(): Convert lists to arrays.
- np.zeros(), np.ones(): Generate arrays with zeros or ones.
- np.arange(), np.linspace(): Create arrays with specified ranges.

Array Operations:

- Element-wise operations: +, -, *, /.
- Dot Product: np.dot() or @ operator.
- Mathematical functions: np.sin(), np.cos(), np.exp(), etc.

Indexing and Slicing:

- Access elements using indices.
- Slice arrays for subarrays.

.

Shape and Reshape:

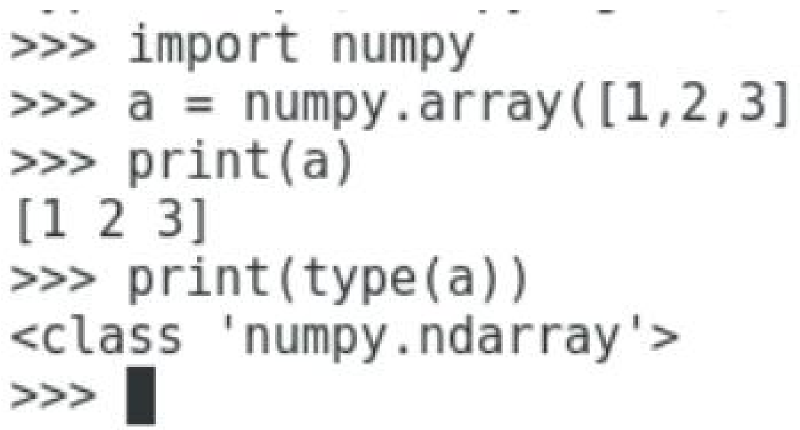
- Check array dimensions using shape.
- Change array shape with reshape().

Statistical Operations:

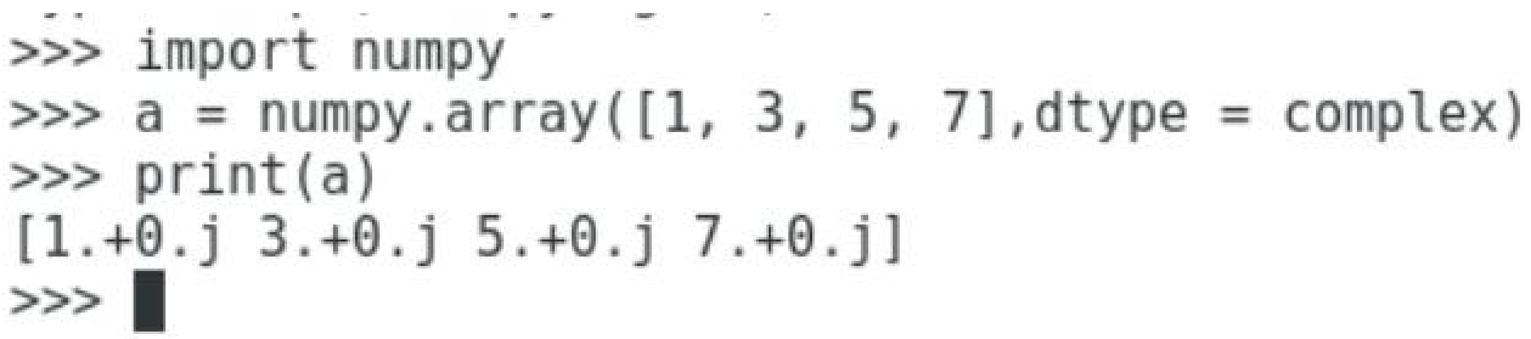
- np.mean(), np.sum(), np.std(), etc.

Broadcasting:

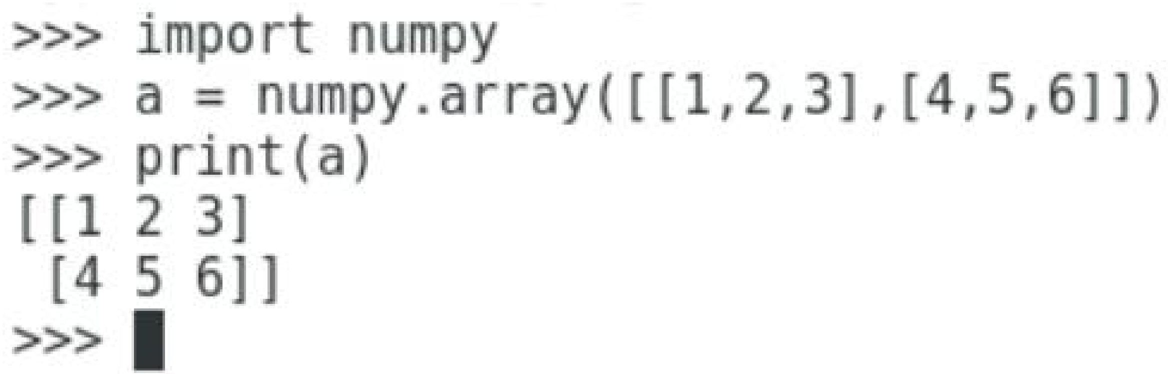
- Implicit element-wise operations on arrays of different shapes



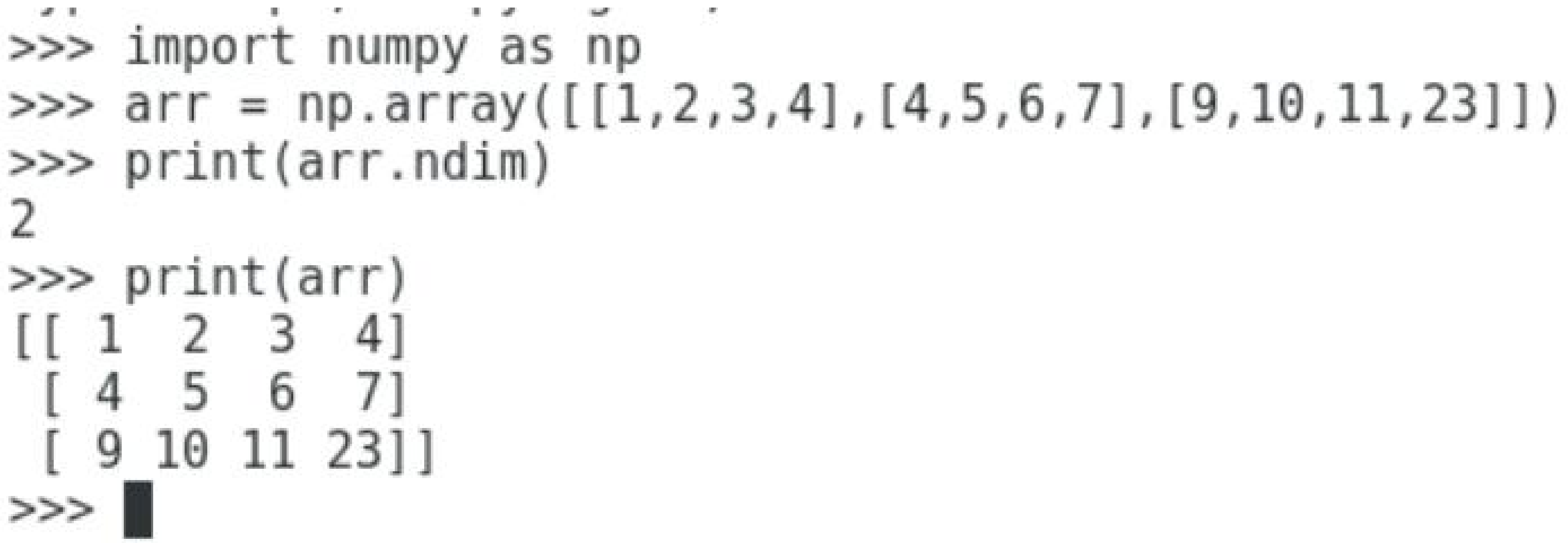
```
>>> import numpy
>>> a = numpy.array([1,2,3]
>>> print(a)
[1 2 3]
>>> print(type(a))
<class 'numpy.ndarray'>
>>>
```



```
>>> import numpy
>>> a = numpy.array([1, 3, 5, 7],dtype = complex)
>>> print(a)
[1.+0.j 3.+0.j 5.+0.j 7.+0.j]
>>>
```



```
>>> import numpy
>>> a = numpy.array([[1,2,3],[4,5,6]])
>>> print(a)
[[1 2 3]
 [4 5 6]]
>>>
```



```
>>> import numpy as np
>>> arr = np.array([[1,2,3,4],[4,5,6,7],[9,10,11,23]])
>>> print(arr.ndim)
2
>>> print(arr)
[[ 1  2  3  4]
 [ 4  5  6  7]
 [ 9 10 11 23]]
>>> 
```

Finding the size of each array element

The itemsize function is used to get the size of each array item. It returns the number of bytes taken by each array element.

```
#finding the size of each item in the array
import numpy as np

a = np.array([[1,2,3]])

print("Each item contains",a.itemsize,"bytes")
```

Output:

```
Each item contains 8 bytes.
```

Finding the data type of each array item

```
import numpy as np

a = np.array([[1,2,3]])

print("Each item is of the type",a.dtype)
```

Output:

```
Each item is of the type int64
```

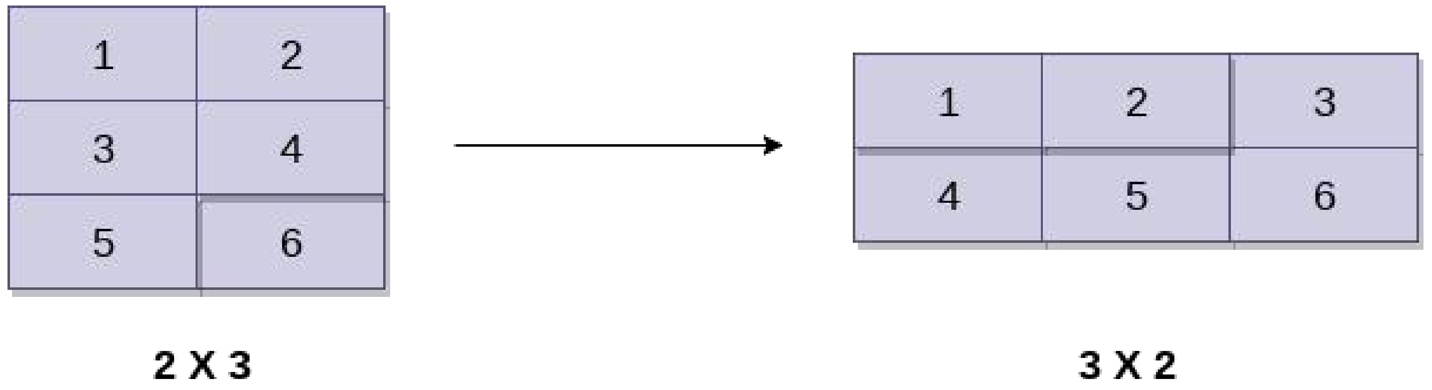
Finding the shape and size of the array

```
import numpy as np
a = np.array([[1,2,3,4,5,6,7]])
print("Array Size:",a.size)
print("Shape:",a.shape)
```

Output:

```
Array Size: 7
Shape: (1, 7)
```

Reshaping the array objects



```
import numpy as np
a = np.array([[1,2],[3,4],[5,6]])
print("printing the original array..")
print(a)
a=a.reshape(2,3)
print("printing the reshaped array..")
print(a)
```

Finding the maximum, minimum, and sum of the array elements

```
import numpy as np

a = np.array([1,2,3,10,15,4])

print("The array:",a)

print("The maximum element:",a.max())

print("The minimum element:",a.min())

print("The sum of the elements:",a.sum())
```

NumPy Array Iteration

NumPy provides an iterator object, i.e., nditer which can be used to iterate over the given array using python standard Iterator interface.

```
import numpy as np
a = np.array([[1,2,3,4],[2,4,5,6],[10,20,39,3]])
print("Printing array:")
print(a);

print("Iterating over the array:")

for x in np.nditer(a):

    print(x,end=' ')
```

1. ***Creating a 1D Array:***

   python

```
import numpy as np

# Using np.array()
array_1d = np.array([1, 2, 3, 4, 5])
```

2. ***Creating a 2D Array:***

   python

```
import numpy as np

# Using np.array() with nested lists
array_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

3. *Creating Arrays with Zeros or Ones:*

```
python
import numpy as np

# Array of zeros
zeros_array = np.zeros(5)  # 1D array of five zeros

# 2D array of ones
ones_array_2d = np.ones((3, 4))  # 3x4 array of ones
```

4. *Creating a Range of Values:*

```
python
import numpy as np

# Using np.arange()
range_array = np.arange(0, 10, 2)  # Array from 0 to 10 (exclusive) with step 2

# Using np.linspace()
linspace_array = np.linspace(0, 1, 5)  # Array of 5 evenly spaced values between 0
```

Pandas

Pandas is defined as an **open-source library** that provides high-performance data manipulation in Python.

The name of Pandas is derived from the word **Panel Data**, which means an **Econometrics from Multidimensional data.**

It is used for data analysis in Python and developed by Wes **McKinney in 2008**.

- Data analysis requires lots of processing, such as **restructuring, cleaning or merging**, etc.

- There are different tools are available for fast data processing, such as **Numpy, Scipy, Cython, and Panda.**

- But we prefer Pandas because working with Pandas is fast, simple and more expressive than other tools.

- **Pandas is built on top of the Numpy package**, means Numpy is required for operating the Pandas.

Python Pandas Data Structure

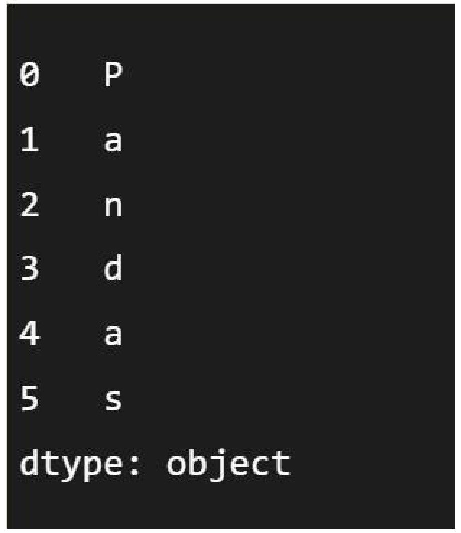
The Pandas provides two data structures for processing the data, i.e., Series and DataFrame.

1) Series

It is defined as a one-dimensional array that is capable of storing various data types. The row labels of series are called the index. We can easily convert the list, tuple, and dictionary into series using "series' method. Data: It can be any list, dictionary, or scalar value.

```
import pandas as pd
import numpy as np
info = np.array(['P','a','n','d','a','s'])
a = pd.Series(info)
print(a)
```

Output



```
0    P
1    a
2    n
3    d
4    a
5    s
dtype: object
```

2)Python Pandas DataFrame

It is a widely used data structure of pandas and works with a two-dimensional array with labeled axes (rows and columns). DataFrame is defined as a standard way to store data and has two different indexes, i.e., row index and column index.

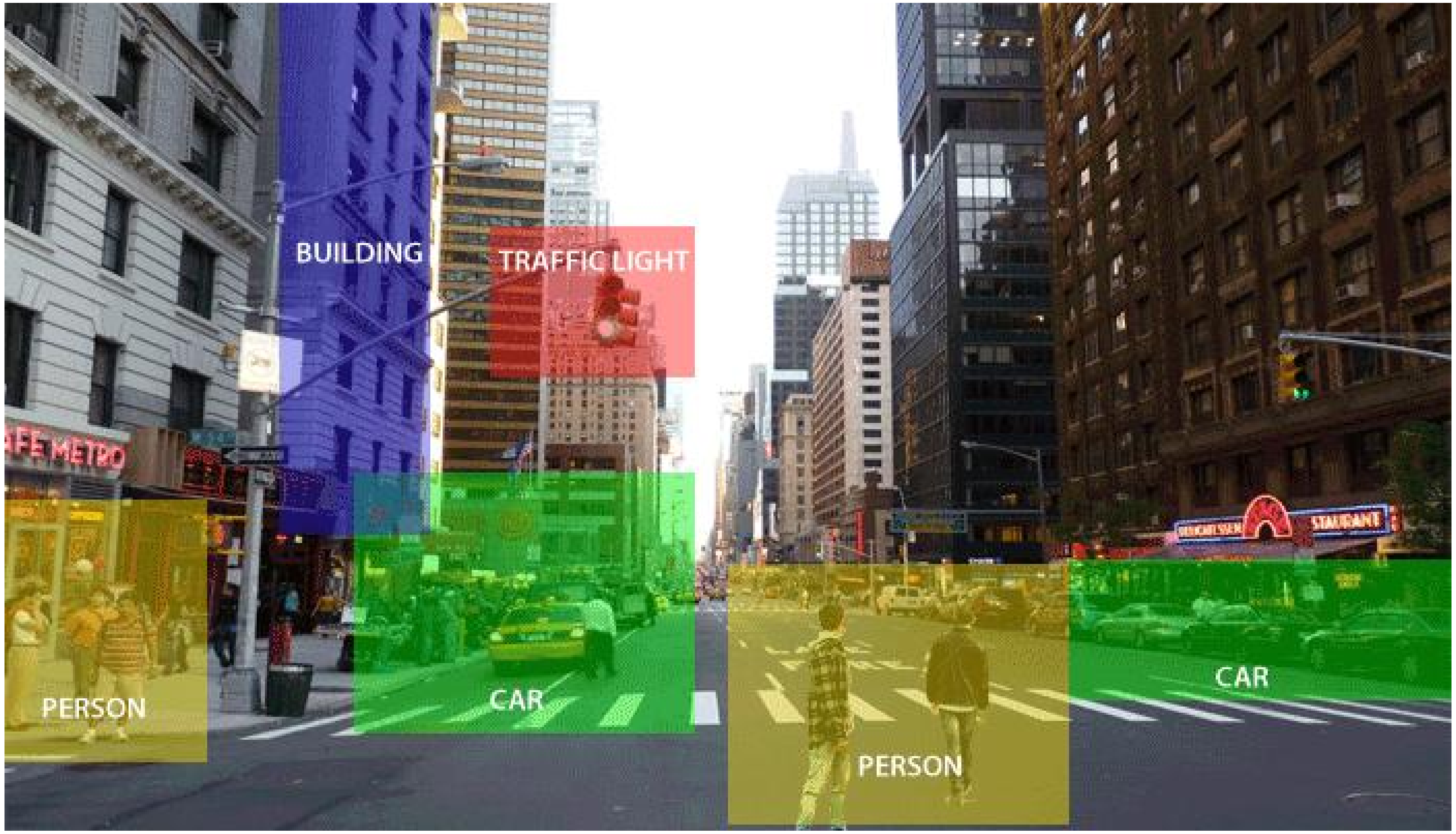
OpenCV?

What is OpenCV?

OpenCV is a Python open-source library, which is used for computer vision in Artificial intelligence, Machine Learning, face recognition, etc.

In OpenCV, the CV is an abbreviation form of a computer vision, which is defined as a field of study that helps computers to understand the content of the digital images such as photographs and videos.

The purpose of **computer vision** is to understand the content of the images. It extracts the description from the pictures, which may be an object, a text description, and three-dimension model, and so on.

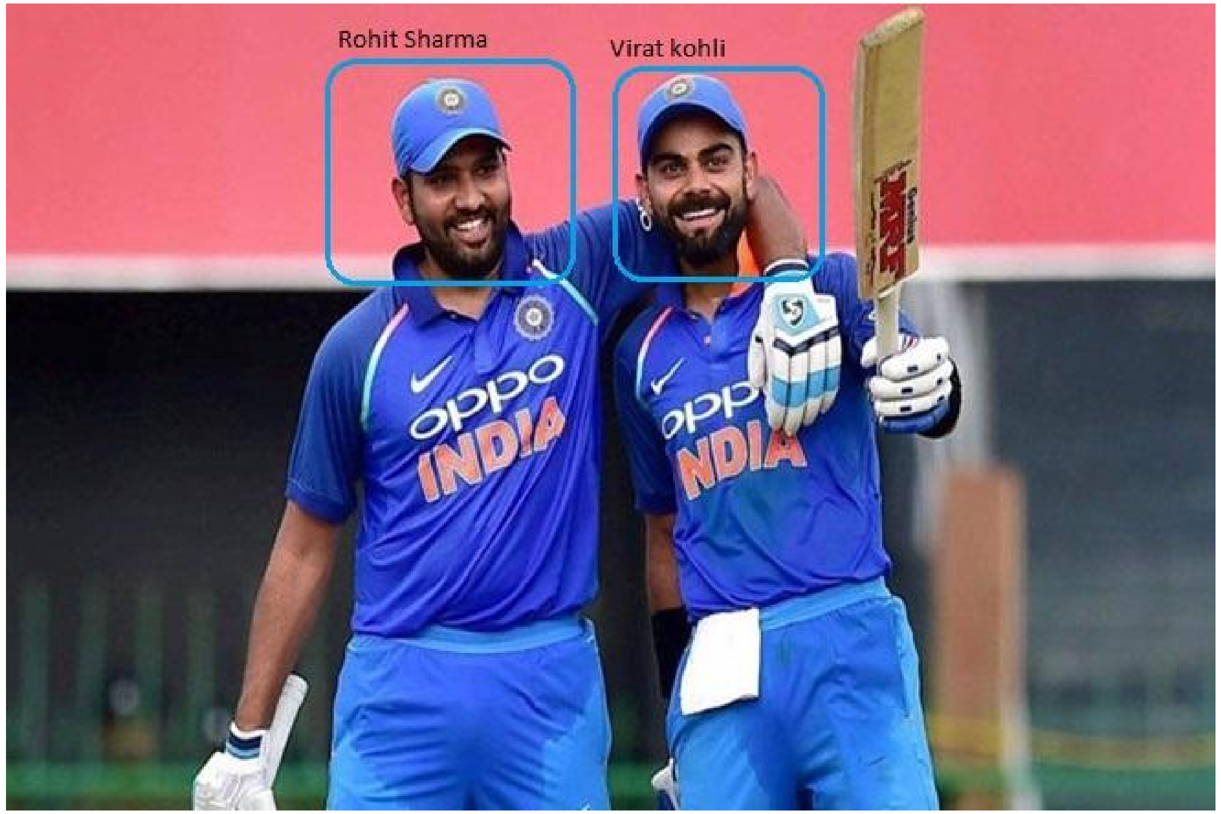


For example, cars can be facilitated with computer vision, which will be able to identify and different objects around the road, such as traffic lights, pedestrians, traffic signs, and so on, and acts accordingly.

Computer vision allows the computer to perform the same kind of tasks as humans with the same efficiency. There are a two main task which are defined below:

Object Classification - In the object classification, we train a model on a dataset of particular objects, and the model classifies new objects as belonging to one or more of your training categories.

Object Identification - In the object identification, our model will identify a particular instance of an object - for example, parsing two faces in an image and tagging one as Virat Kohli and other one as Rohit Sharma.



History

OpenCV stands for Open Source Computer Vision Library, which is widely used for image recognition or identification. It was officially launched in 1999 by Intel. It was written in C/C++ in the early stage, but now it is commonly used in Python for the computer vision as well.

The first alpha version of OpenCV was released for the common use at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and between 2001 and 2005, five betas were released. The first 1.0 version was released in 2006.

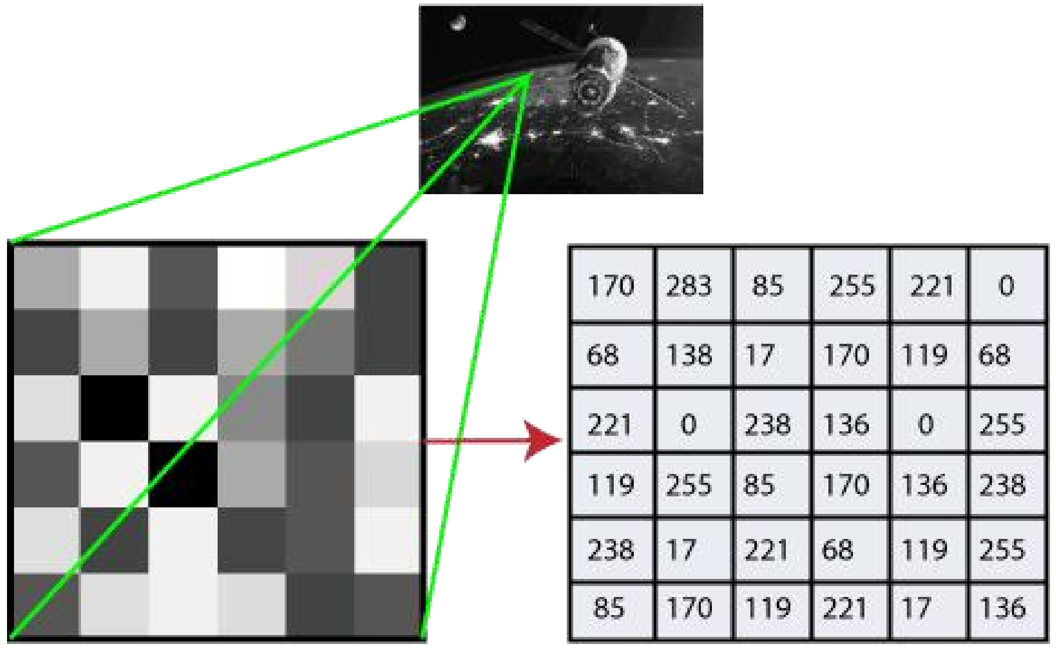
How does computer recognize the image?

Human eyes provide lots of information based on what they see.

Machines are facilitated with seeing everything, convert the vision into numbers and store in the memory.

Here the question arises how computer convert images into numbers. So the answer is that the pixel value is used to convert images into numbers.

A pixel is the smallest unit of a digital image or graphics that can be displayed and represented on a digital display device.



170	283	85	255	221	0
68	138	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

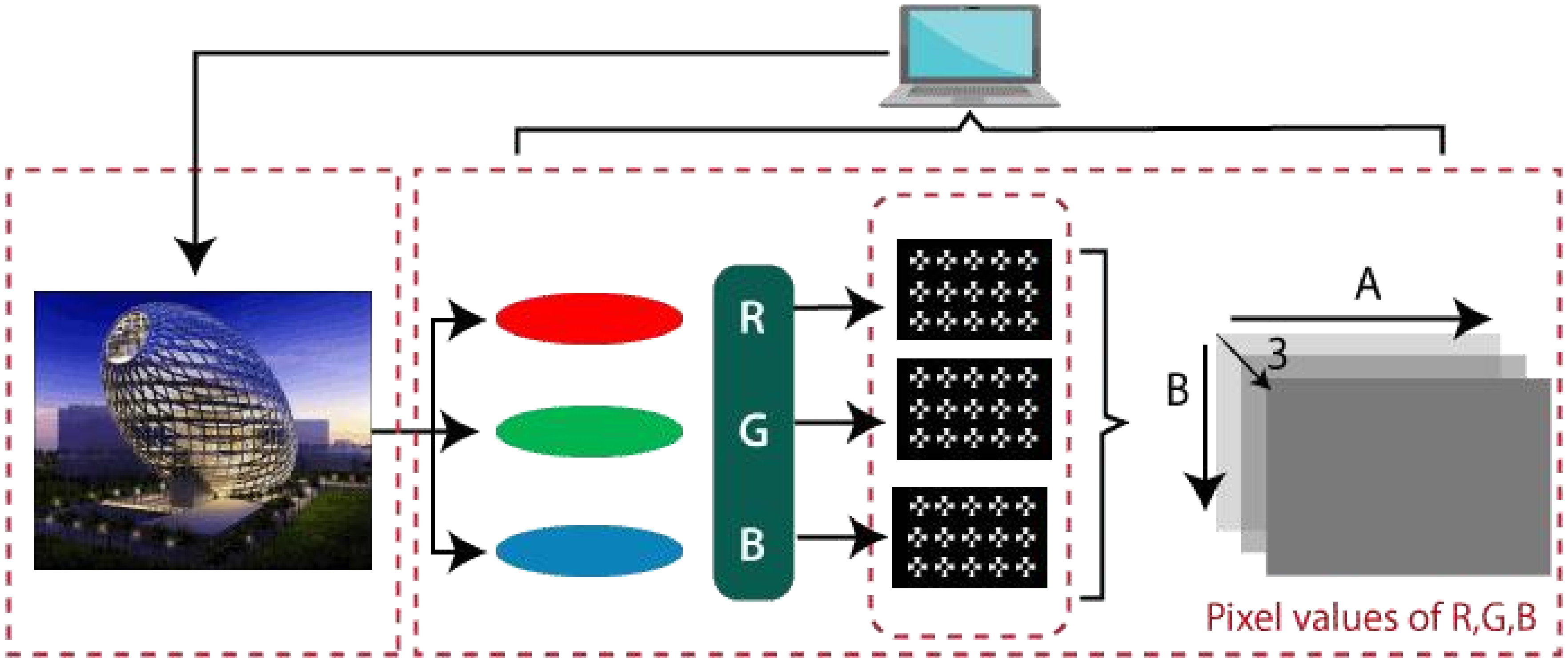
There are two common ways to identify the images:

1. Grayscale

Grayscale images are those images which contain only two colors black and white. The contrast measurement of intensity is black treated as the weakest intensity, and white as the strongest intensity. When we use the grayscale image, the computer assigns each pixel value based on its level of darkness.

2. RGB

An RGB is a combination of the red, green, blue color which together makes a new color. The computer retrieves that value from each pixel and puts the results in an array to be interpreted.



Install OpenCV in the Windows via pip

```
pip install opencv-python
```

OpenCV imread function

The imread() function loads image from the specified file and returns it. The syntax is:

cv2.imread(filename[,flag])

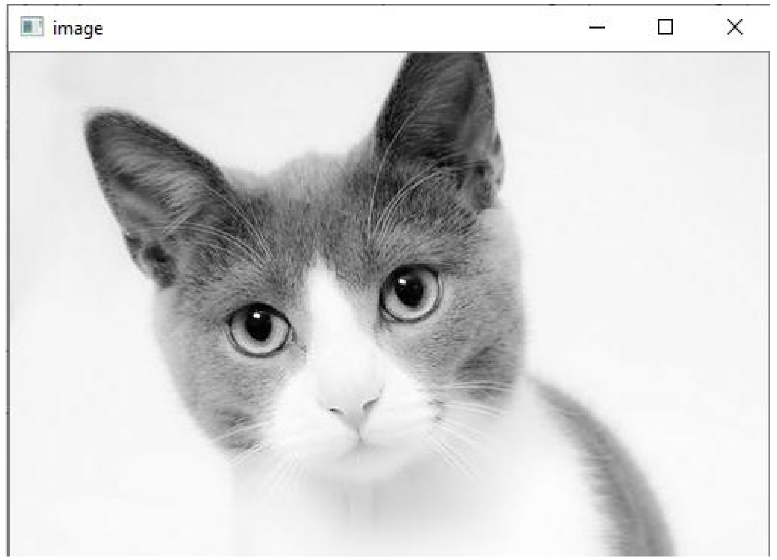
Parameters:
filename: Name of the file to be loaded

flag: The flag specifies the color type of a loaded image:

CV_LOAD_IMAGE_ANYDEPTH - If we set it as flag, it will return 16-bits/32-bits image when the input has the corresponding depth, otherwise convert it to 8-BIT.
CV_LOAD_IMAGE_COLOR - If we set it as flag, it always return the converted image to the color one.
C V_LOAD_IMAGE_GRAYSCALE - If we set it as flag, it always convert image into the grayscale.

OpenCV imread function

```
#importing the opencv module
import cv2

# using imread('path') and 0 denotes read as  grayscale image

img = cv2.imread(r'C:\Users\SRM\cat.jpeg',1)

#This is using for display the image

cv2.imshow('image',img)

cv2.waitKey(3) # This is necessary to be required so that the image doesn't close
immediately.
#It will run continuously until the key press.
cv2.destroyAllWindows()
```



OpenCV Save Images

OpenCV imwrite() function is used to save an image to a specified file. The file extension defines the image format. The syntax is the following:

cv2.imwrite(filename, img[,params])

```
import cv2

# read image as grey scale
img = cv2.imread(r'C:\Users\SRM\cat.jpeg', 1)

# save image
status = cv2.imwrite(r'C:\Users\SRM\cat.jpeg',  img)

print("Image written to file-system : ", status)
```

OpenCV Basic Operation on Images

Accessing and Modifying pixel values

We can retrieve a pixel value by its row and column coordinates. It returns an array of blue, green, red values of the BGR image. It returns the corresponding intensity for the grayscale image. First, we need to load the BGR image.

```
import numpy as np
import cv2
img = cv2.imread("C:\Users\SRM\cat.jpeg",1)
pixel = img[100,100]
print(pixel)
```

Output:

```
[190 166 250]
```

Accessing Image Properties

```
height = img.shape[0]
width = img.shape[1]
channels = img.shape[2]
size1 = img.size
```

Image ROI (Region of Interest)

Sometimes, we need to work with some areas of the image. As we discuss in the previous tutorial face detection is over the entire picture. When a face is obtained, we select only the face region and search for eyes inside it instead of searching the whole image. It enhances accuracy and performance because eyes are always on the face and don't need to search the entire image.

Change in Image color

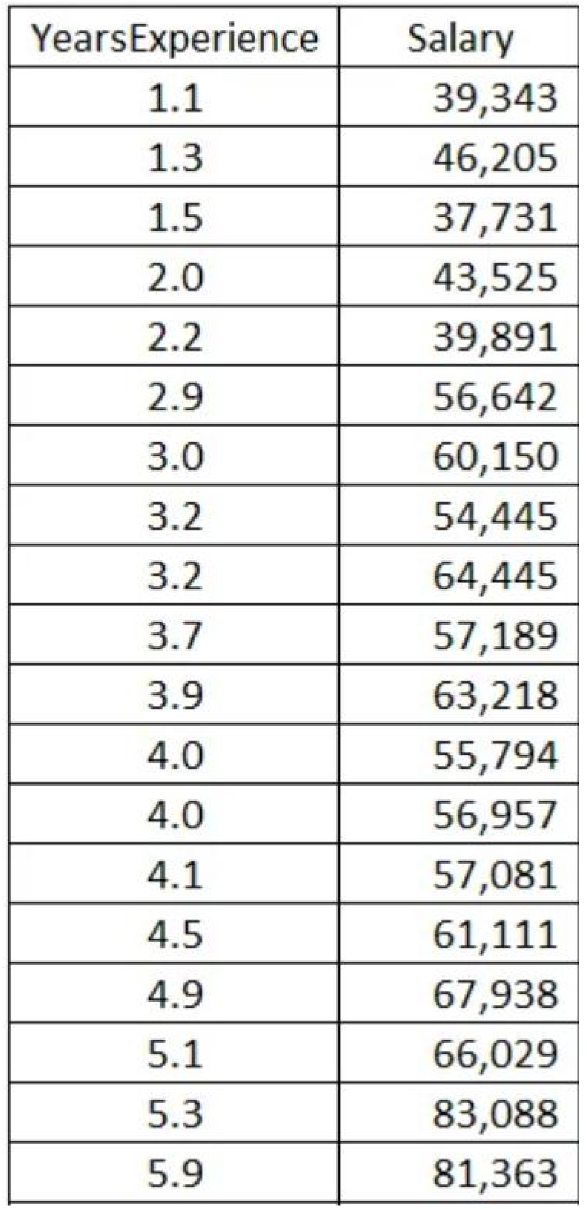
OpenCV **cvtColor**

The cvtColor is used to convert an image from one color space to another. The syntax is following:

```
cv2.cvtColor(src, dst, code)
```

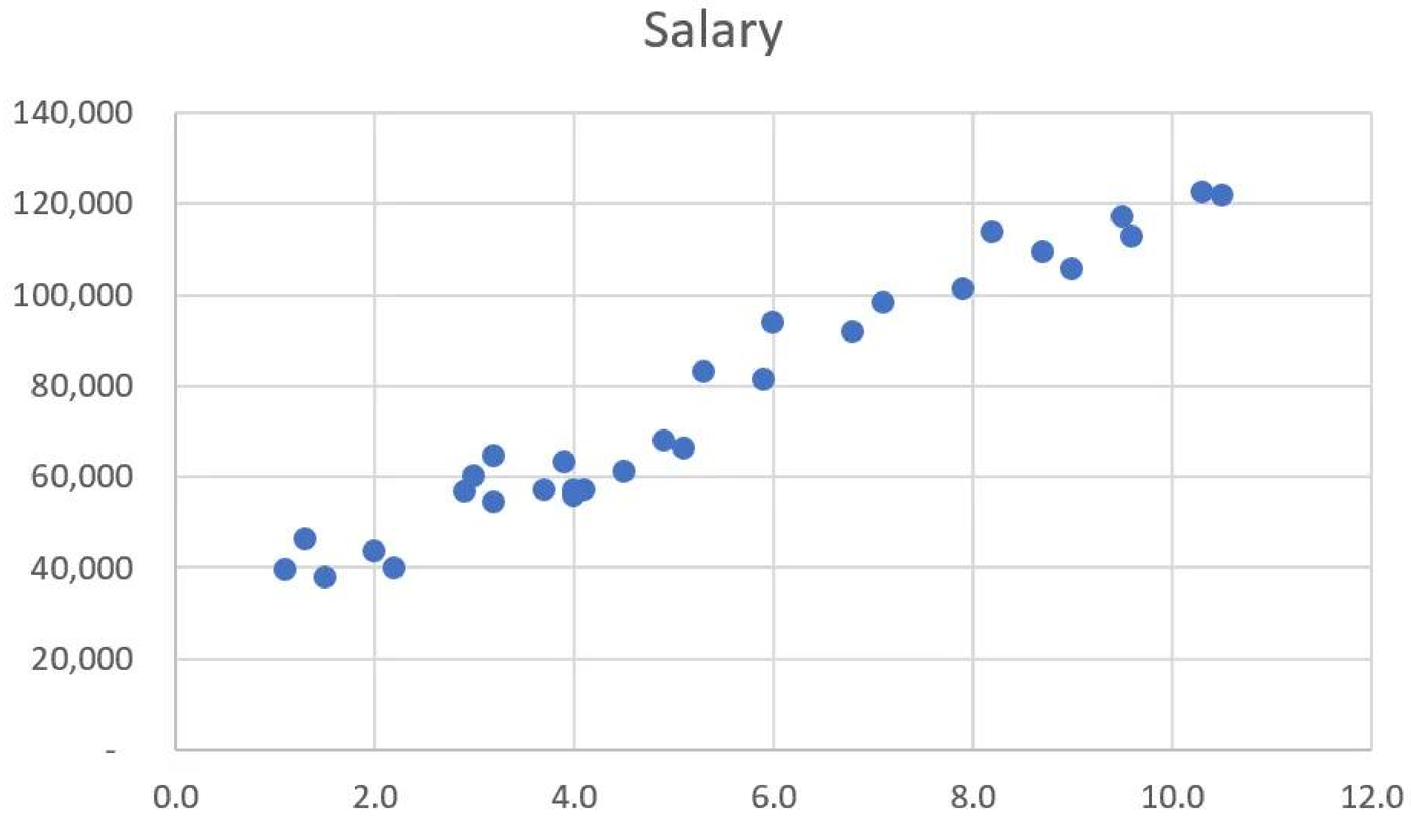
```
image = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY )
```

Simple Linear Regression with Python



YearsExperience	Salary
1.1	39,343
1.3	46,205
1.5	37,731
2.0	43,525
2.2	39,891
2.9	56,642
3.0	60,150
3.2	54,445
3.2	64,445
3.7	57,189
3.9	63,218
4.0	55,794
4.0	56,957
4.1	57,081
4.5	61,111
4.9	67,938
5.1	66,029
5.3	83,088
5.9	81,363

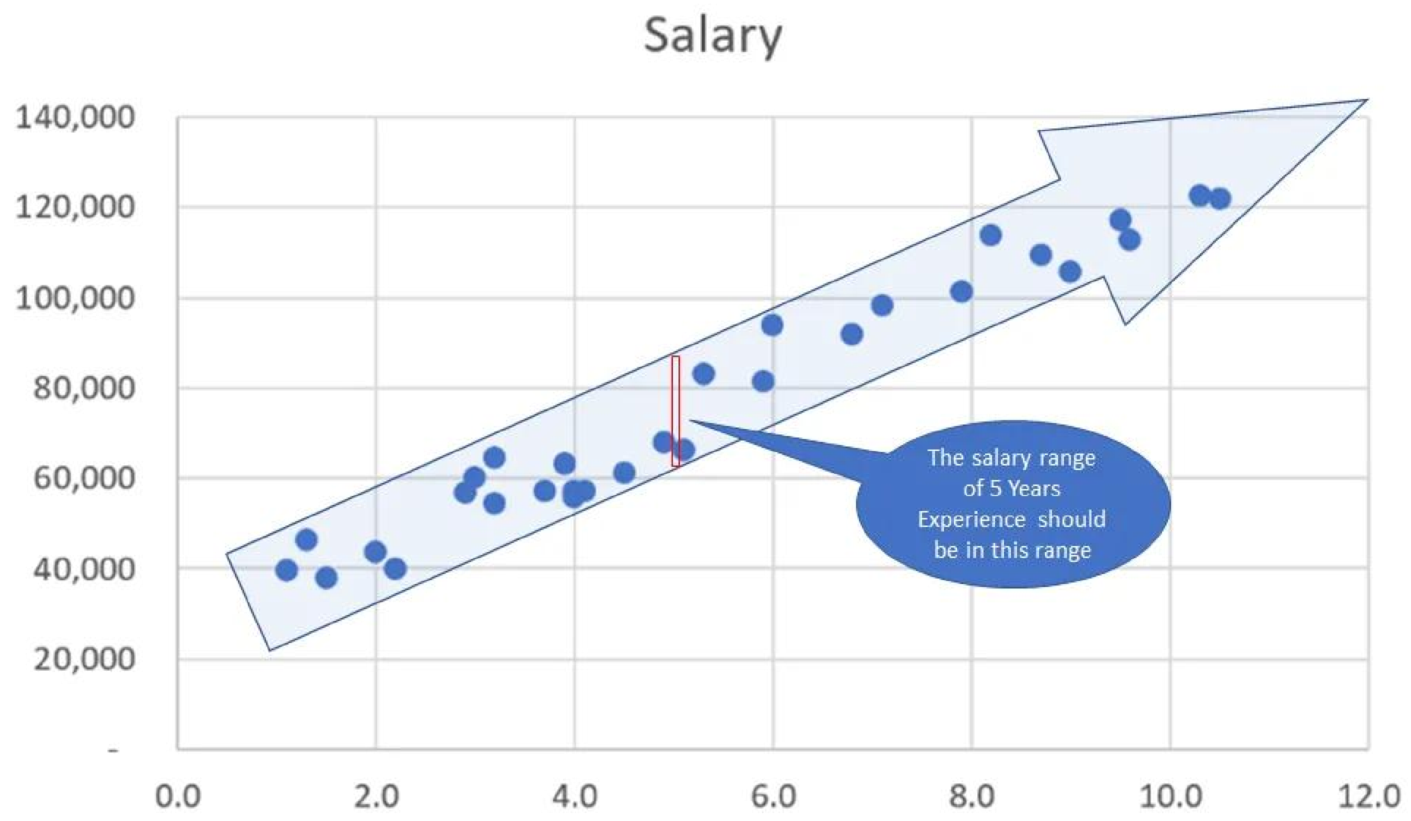
"The scenario is you are a HR officer, you got a candidate with 5 years of experience. Then what is the best salary you should offer to him?"



all the observations are not in a line. It means we cannot find out the equation to calculate the (y) value.

Look at the Scatter Plot again. Do you see it?

All the points is not in a line BUT they are in a line-shape! ***It's linear!***



Based on our observation, we can guess that the salary range of 5 Years Experience should be in the red range.

Of course, we can offer to our candidate any number in that red range.

But how to pick the best number for him? It's time to use Machine Learning to predict the best salary for our candidate.

Linear Regression with Python

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

# Importing the dataset

dataset = pd.read_csv('salary_data.csv')
X = dataset.iloc[:, :-1].values #get a copy of dataset exclude last column
y = dataset.iloc[:, 1].values #get array of dataset in column 1st
```

dataset: the table contains all values in our csv file

X: the first column which contains Years Experience array

y: the last column which contains Salary array

Next, we have to split our dataset (total 30 observations) into 2 sets:

training set which used for training and test set which used for testing:

```
# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=0)
```

test_size=1/3: we will split our dataset (30 observations) into 2 parts (training set, test set) and the ratio of test set compare to dataset is 1/3 (10 observations will be put into the test set. You can put it 1/2 to get 50% or 0.5, they are the same. We should not let the test set too big; if it's too big, we will lack of data to train. Normally, we should pick around 5% to 30%.

random_state: this is the seed for the random number generator. We can put an instance of the RandomState class as well. If we leave it blank or 0, the RandomState instance used by np.random will be used instead.

We already have the train set and test set, now we have to build the Regression Model:

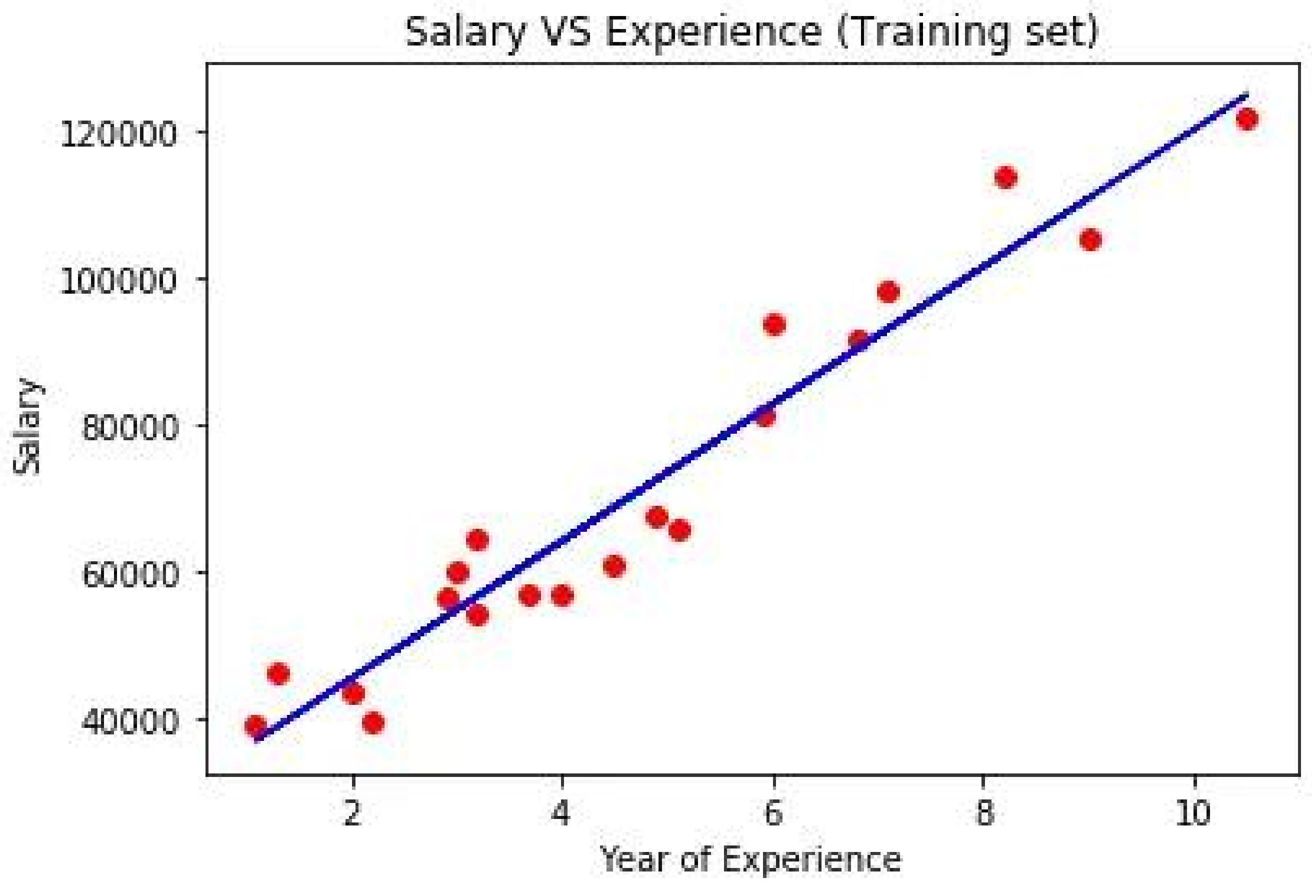
```
# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

regressor = LinearRegression(): our training model which will implement the Linear Regression.

regressor.fit: in this line, we pass the X_train which contains value of Year Experience and y_train which contains values of particular Salary to form up the model. This is the training process.

Let’s visualize our training model and testing model:

```
plt.scatter(X_train, y_train, color='red')
```



```
# Predicting the result of 5 Years Experience
y_pred = regressor.predict(5)
```