

## Unit V

Object Oriented Programming: Introduction to Classes, Objects and Methods, Standard Libraries. Overview of stacks and queues. Overview of packages:networkx,matplotlib.pyplot, numpy.

---

# OOPs Concepts

Like other general-purpose programming languages, Python is also an object-oriented language since its beginning. It allows us to develop applications using an Object-Oriented approach. In Python, we can easily create and use classes and objects.

An object-oriented paradigm is to design the program using classes and objects. The object is related to real-world entities such as book, house, pencil, etc. The oops concept focuses on writing the reusable code. It is a widespread technique to solve the problem by creating objects.

Major principles of object-oriented programming system are given below.

- Class
- Object
- Method
- Inheritance
- Polymorphism
- Data Abstraction
- Encapsulation

## Class

The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. For example: if you have an employee class, then it should contain an attribute and method, i.e. an email id, name, age, salary, etc.

### Some points on Python class:

- Classes are created by keyword class.
- Attributes are the variables that belong to a class.
- Attributes are always public and can be accessed using the dot (.) operator. Eg.: Myclass.Myattribute

## Object

The object is an entity that has state and behavior. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc.

Everything in Python is an object, and almost everything has attributes and methods. All functions have a built-in attribute `__doc__`, which returns the docstring defined in the function source code.

An Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with *actual values*. It's not an idea anymore, it's an actual dog, like a dog of breed pug who's seven years old. You can have many dogs to create many different instances, but without the class as a guide, you would be lost, not knowing what information is required.

<pre>class car:      def __init__(self,modelname, year):          self.modelname = modelname          self.year = year      def display(self):          print(self.modelname,self.year)  c1 = car("Toyota", 2016)  c1.display()</pre>	Toyota 2016
---	-------------

## Method

The method is a function that is associated with an object. In Python, a method is not unique to class instances. Any object type can have methods.

## Python Standard Library

The standard library is a huge collection of all sort of utilities, ranging from math utilities to debugging to creating graphical user interfaces.

Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

Some of the important modules are:

- `math` for math utilities
- `re` for regular expressions
- `json` to work with JSON
- `datetime` to work with dates

- `sqlite3` to use SQLite
- `os` for Operating System utilities
- `random` for random number generation
- `statistics` for statistics utilities
- `requests` to perform HTTP network requests
- `http` to create HTTP servers
- `urllib` to manage URLs

Python libraries that used in Machine Learning are:

- Numpy
- Scipy
- Scikit-learn
- Theano
- TensorFlow
- Keras
- PyTorch
- Pandas
- Matplotlib

## Overview of packages

**NumPy** is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.

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

There are the following advantages of using NumPy for data analysis.

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.

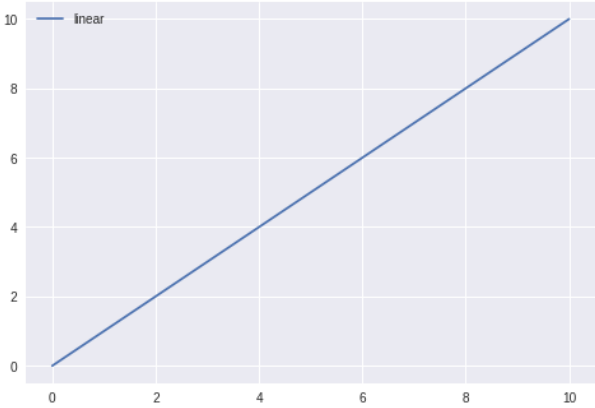
# Python program using NumPy # for some basic mathematical # operations import numpy as np  # Creating two arrays of rank 2 x = np.array([[1, 2], [3, 4]]) y = np.array([[5, 6], [7, 8]])  # Creating two arrays of rank 1	219  [29 67]  [[19 22] [43 50]]
---	--

<pre> v = np.array([9, 10]) w = np.array([11, 12])  # Inner product of vectors print(np.dot(v, w), "\n")  # Matrix and Vector product print(np.dot(x, v), "\n")  # Matrix and matrix product print(np.dot(x, y)) </pre>	
---	--

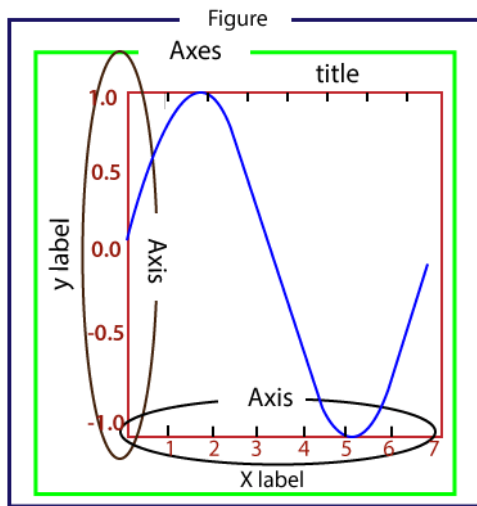
**Matplotlib** is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar charts, etc,

**Matplotlib** is a plotting library for creating static, animated, and interactive visualizations in Python. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface toolkits like Tkinter, awxPython, etc.

**Matplotlib** is a Python library which is defined as a multi-platform data visualization library built on Numpy array. It can be used in python scripts, shell, web application, and other graphical user interface toolkit.

<pre> # Python program using Matplotlib # for forming a linear plot  # importing the necessary packages and modules import matplotlib.pyplot as plt import numpy as np  # Prepare the data x = np.linspace(0, 10, 100)  # Plot the data plt.plot(x, x, label='linear')  # Add a legend plt.legend()  # Show the plot plt.show() </pre>	
--	--

## General Concept of Matplotlib



**Figure:** It is a whole figure which may hold one or more axes (plots). We can think of a Figure as a canvas that holds plots.

**Axes:** A Figure can contain several Axes. It consists of two or three (in the case of 3D) Axis objects. Each Axes is comprised of a title, an x-label, and a y-label.

**Axis:** Axes are the number of line like objects and responsible for generating the graph limits.

**Artist:** An artist is the all which we see on the graph like Text objects, Line2D objects, and collection objects. Most Artists are tied to Axes.

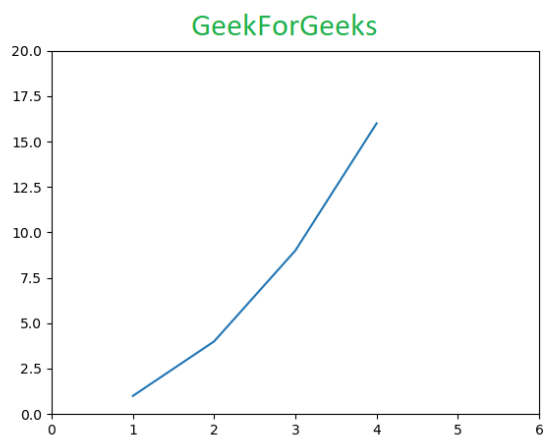
## Pyplot

**Pyplot** is a Matplotlib module which provides a MATLAB-like interface. Matplotlib is designed to be as usable as MATLAB, with the ability to use Python and the advantage of being free and open-source. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. The various plots we can utilize using Pyplot are **Line Plot, Histogram, Scatter, 3D Plot, Image, Contour, and Polar**.

# Python program to show plot function

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.axis([0, 6, 0, 20])
plt.show()
```



# NetworkX

*NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.”*

NetworkX is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. Pygraphviz is a Python interface to the Graphviz graph layout and visualization package.

1. NetworkX allows us to create, manipulate, and study structure, functions and dynamics of complex networks.
2. It supports data structures for graphs, digraphs, and multigraphs
3. It has numerous standard graph algorithms.

`$ pip install networkx`

## Python NetworkX

- NetworkX is suitable for real-world graph problems and is good at handling big data as well.
- As the library is purely made in python, this fact makes it highly scalable, portable and reasonably efficient at the same time.
- It is open source and released under 3-clause BSD License.

## Why NetworkX?

NetworkX gives you a lot of reasons to go with it. Following are some features of NetworkX that makes it a package to go with:

- NetworkX has numerous standard graph algorithms
- It supports data structures for graphs, digraphs, and multigraphs
- It provides various network structure and measures for analysis
- Making classic/random graphs and synthetic networks is much easier using generators provided in the package
- Nodes in your network or graph can be absolutely anything, be it images, XML data or anything else
- Edges also can hold arbitrary data like timestamp and weight
- It has been well tested with about 90% code coverage

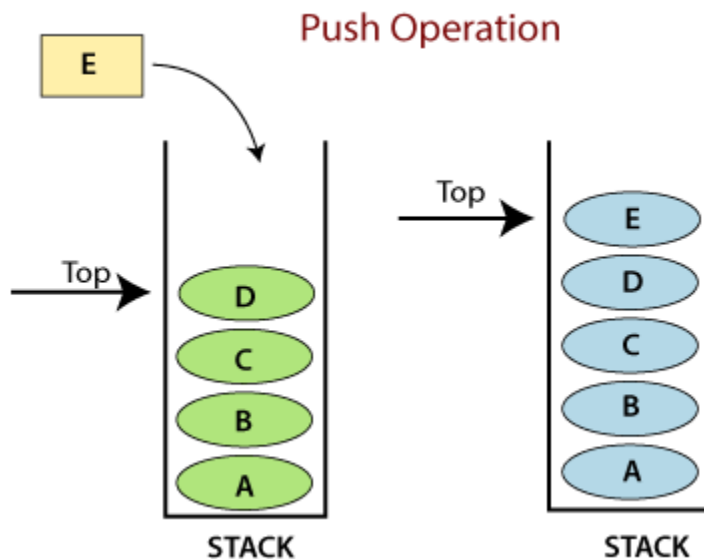
# Stack and Queue

Data structure organizes the storage in computers so that we can easily access and change data. Stacks and Queues are the earliest data structure defined in computer science. A simple Python list can act as a queue and stack as well. A queue follows FIFO rule (First In First Out) and used in programming for sorting. It is common for stacks and queues to be implemented with an array or linked list.

## Stack

A Stack is a data structure that follows the LIFO (Last In First Out) principle. To implement a stack, we need two simple operations:

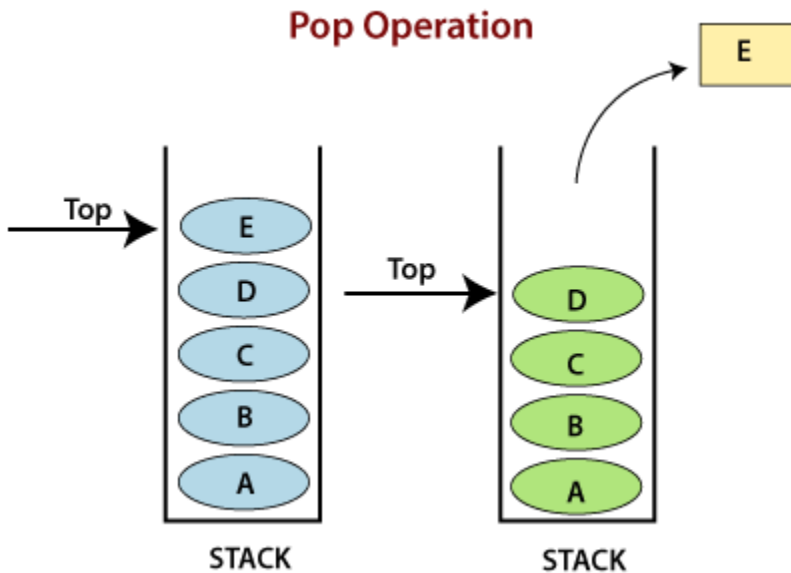
- **push** - It adds an element to the top of the stack.
- **pop** - It removes an element from the top of the stack.



### Operations:

- **Adding** - It adds the items in the stack and increases the stack size. The addition takes place at the top of the stack.

- **Deletion** - It consists of two conditions, first, if no element is present in the stack, then underflow occurs in the stack, and second, if a stack contains some elements, then the topmost element gets removed. It reduces the stack size.
- **Traversing** - It involves visiting each element of the stack.



#### Characteristics:

- Insertion order of the stack is preserved.
- Useful for parsing the operations.
- Duplicacy is allowed.

<pre># Code to demonstrate Implementation of # stack using list  x = ["Python", "C", "Android"]  x.push("Java")  x.push("C++")  print(x)  print(x.pop())  print(x)  print(x.pop())  print(x)</pre>	<pre>['Python', 'C', 'Android', 'Java', 'C++'] C++ ['Python', 'C', 'Android', 'Java'] Java ['Python', 'C', 'Android']</pre>
--	---

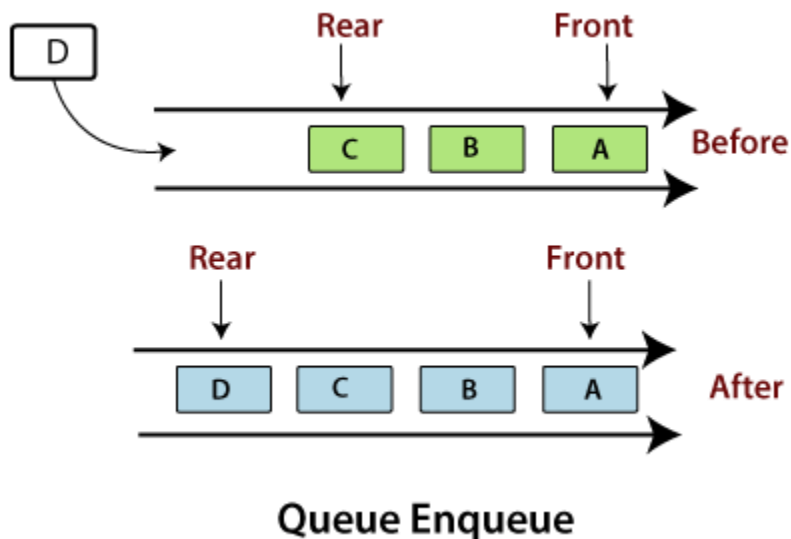


# Queue

A Queue follows the First-in-First-Out (FIFO) principle. It is opened from both the ends hence we can easily add elements to the back and can remove elements from the front.

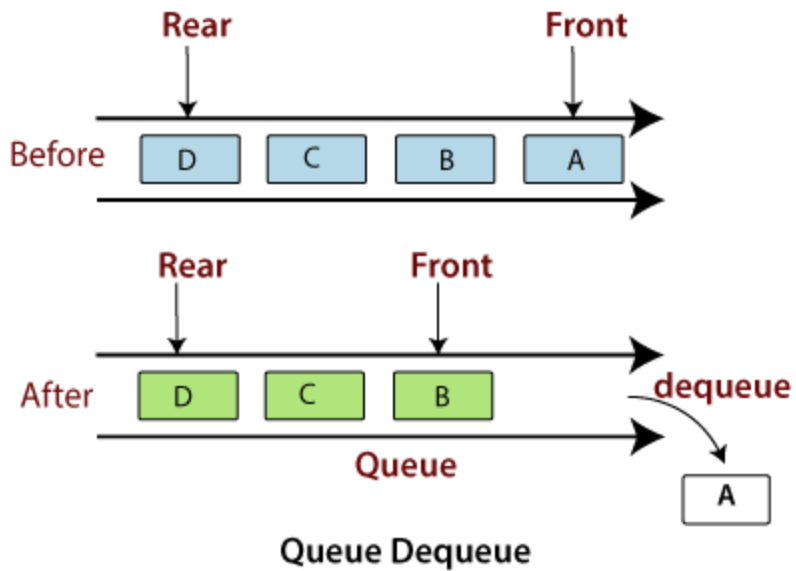
To implement a queue, we need two simple operations:

- **enqueue** - It adds an element to the end of the queue.
- **dequeue** - It removes the element from the beginning of the queue.



## Operations on Queue

- **Addition** - It adds the element in a queue and takes place at the rear end, i.e., at the back of the queue.
- **Deletion** - It consists of two conditions - If no element is present in the queue, Underflow occurs in the queue, or if a stack contains some elements then element present at the front gets deleted.
- **Traversing** - It involves to visit each element of the queue.



### Characteristics

- Insertion order of the queue is preserved.
- Duplicacy is allowed.
- Useful for parsing CPU task operations.

<pre> import queue # Queue is created as an object 'L' L = queue.Queue(maxsize=10) # Data is inserted in 'L' at the end using put() L.put(9) L.put(6) L.put(7) L.put(4) # get() takes data from # from the head # of the Queue print(L.get()) print(L.get()) print(L.get()) print(L.get()) </pre>	<pre> 9 6 7 4 </pre>
---	----------------------