1) **Object-Oriented programming** is a concept which is used in python to ease the way of doing complex programs. In this method, we define a class which consists of variables and methods to perform operations. Using the single class defined, the operations with various values can be done. There is no need to define every operation each time we perform.

   For eg, consider the construction of a house. We need plan/sketch to construct a house. By using this, we give the exact measurement for the house and will build the house. Here the sketch of the house can be considered as class and the object we created from the plan(class) is the house.

2) **Classes:** Classes are defined as a blue print which defines all the nature of an object. It consists of attributes and methods (a special kind of function) which helps in performing the operations.
   Eg:
```
class Student:
    def __init__(self,roll_no,Name):
        self.name = Name
        self.roll_no = roll_no

    def display(self):
        print("Name of student: ",self.name)
        print("Roll no. of student: ",self.roll_no)
```

   Objects: Objects are created from the class. They are created by giving the attributes to the class. The method of creating object is called instantiation.
   Eg: stu1 = Student(1,"Akhil")

3) Inheritance:

In object-oriented programming there may be different classes which has got same attributes. So while doing a complex programming, we don't want to write that again and again. So we can add those same attributes in a single class and other classes can inherit the contents of that class. This is called Inheritance. The class from which the attributes are inherited is called the parent class and class which inherit the attributes from another class is called child class.

Types of inheritances:

- **Single level inheritance**: It is the inheritance which is done in a single level. i.e, there will be a single parent class.

  Eg:
```
class Student:
    def __init__(self,roll_no,Name):
        self.name = Name
        self.roll_no = roll_no

    def display(self):
        print("Name of student: ",self.name)
        print("Roll no. of student: ",self.roll_no)

class Marks(Student):

    def screen(self):
        print("Name: ",self.name)
stu1 = Marks(1,"Akhil")
```

```
stu1.screen()
```

- **Multilevel inheritance**: In this type, a child class is again inherited by another class.

```
Eg: class A:
    def __init__(self,a):
        self.a = a


class B(A):


    def __init__(self,a,b):
        A.__init__(self,a)
        self.b = b


class C(B):

    def show(self):
        print(self.a)
        print(self.b)

new = C(1,2)
new.show()
```

- **Multiple Inheritance**: In this type, a child class can inherit the attributes of multiple classes.

```
Eg: class A:
    def __init__(self,a):
        self.a = a


class B:
    def __init__(self,b):
        self.b = b


class C(A,B):

    def __init__(self,a,b):
        A.__init__(self,a)
        B.__init__(self,b)

    def show(self):
        print(self.a)
        print(self.b)

new = C(1,2)
new.show()
```

4) When a parent class and child class have the same method inside it, the method in child class will override the method in parent class to perform some special operations. This is called **method overriding.**

Eg:
```
class Parent:
    def __init__(self):
        self.message = "Inside parent"

    def show(self):
        print(self.message)

class Child(Parent):
    def __init__(self):
        self.message = "Inside child"

    def show(self):
        print(self.message)

obj1 = Parent()
obj2 = Child()

obj1.show()
obj2.show()
```

5) **Operator overloading**: After instantiating objects using the class defined by the user, performing arithmetic or logical operations in the objects may lead to an error. This is because the python doesn't know how to use the operator in the object. But when using the predefined objects, arithmetic and logical operations can be done without any error because the operation is predefined inside the class already. So when we create class we have to define the operation that has to be done by the operator. Also we can't create a new operator but we can define the operations for the existing operator.

Eg:
```
class total:

    def __init__(self,a,b):
        self.a = a
```

```
        self.b = b

    def __str__(self):
        return "{}{}".format(self.a,self.b)

    def __add__(self,other):
        a = self.a + other.a
        b= self.b + other.b
        return a,b

p1 = total(1,2)
p2 = total(3,4)
p3 = p1 + p2
print(p3)
```

6) While creating class, user can hide an attribute or method from displaying. It is done by putting 2 underscores before the attributes or methods the user want to hide.
7) Numpy is a library in python which is used to perform various tasks with array. For installing the Numpy library, the following syntax is used.
   Pip install numpy
8) For creating ndarray object, first we have to import the numpy. Then using array() function, ndarray objects can be created.
   import numpy as np
   arr = np.array(data)
9) Numpy array copy means it is creating a copy of the array with a different memory location. So when we change the value in copy array, it will not affect the original array
   Numpy array view means its just a view of the array. So any change in the view array will affect the original array.
10) A) numpy.shape()
    b) numpy.ndim()
    c)np.reshape(arr)
    d)for i in np.nditer(array)
    e)array[index]
      array[start:stop:step]
11) To install pandas use the following statement *pip install pandas*
12) Numpy is dealing with arrays which is homogenous in nature where as pandas deals with heterogenous elements.
13) **Series** is a kind of an array structure which deals with same kind of data types. Also the size of the series is immutable but the elements are mutable. In case of **dataframes**, they deals with different kind of datatypes. The size and elements of a dataframe can be changed anytime, i.e, its mutable

14) import pandas as pd
    lst = [1,2,3]
    p = pd.Series(lst)
    print(p)
15) dict = {'name':['aswin','akhil'],'age':[23,25]}
    df = pd.DataFrame(dict)
    print(df)
16) import pandas as pd
    file = pd.read_excel("F:/Python/Book1.xlsx")
    df = pd.DataFrame(file)
    print(df)
17) import pandas as pd
    file = pd.read_csv("F:/Python/Book1.csv")
    df = pd.DataFrame(file)
    print(df)
18) my sql queries
    1) create database database_name
    2) create table table_name(col1 datatype property,col2…..)
    3) insert into table_name values(val1,val2…)
    4) alter table table_name drop column column_name
    5) select * from table_name
19)
```
a) select * from employees inner join salary on employees.ID = salary.Emp_ID
   select * from employees left join salary on employees.ID = salary.Emp_ID
   select * from employees right join salary on employees.ID = salary.Emp_ID
b) select * from employees where ID between 4 and 10
c) select * from employees where ID > 3
d) select * from employees where Name like "A%"
e select * from employees where Name = "Abilna" or Name = "Binoy" or Name = "Raju"
f)
g) update employees set Name = "Trivandrum" where ID = 10
h) select * from employees order by Name
i) select * from salary order by salary
```