

## Experiment No: 1

**Aim:** Write python program to implement

- a. basic string operations
- b. basic list operations
- c. tuple operations
- d. dictionary operations
- e. set operations

**Software Used:** Python

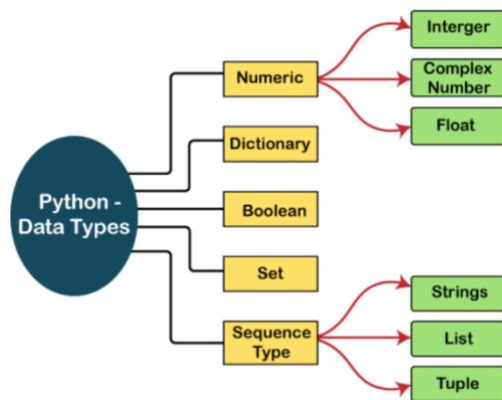
### Theory:

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

### Numbers

Integer value can be any length such as integers 10, 2, 29, -20, -150 etc. Python has no restriction on the length of an integer. Float is used to store

floating-point numbers like 1.9, 9.902, 15.2, etc. It is accurate up to 15 decimal points. A complex number contains an ordered pair, i.e.,  $x + iy$  where  $x$  and  $y$  denote the real and imaginary parts, respectively. The complex numbers like 2.14j, 2.0 + 2.3j, etc.



### Strings

The string can be defined as the sequence of characters represented in the quotation marks. In Python, we can use single, double, or triple quotes to define a string. String handling in Python is a straightforward task since Python provides built-in functions and

operators to perform operations in the string. In the case of string handling, the operator `+` is used to concatenate two strings as the operation `"hello"+"python"` returns `"hello python"`. The operator `*` is known as a repetition operator as the operation `"Python" * 2` returns `'Python Python'`

#### Basic String Operations

1. Slicing
2. Reversing
3. Deleting a string
4. Concatenation
5. Repetition
6. Finding the length of the string
7. capitalizes the first character of the string
8. convert all the characters of a string to Lower/Upper case
9. Removes all trailing/leading whitespace of a string

### Lists

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.

#### Basic List Operations

1. Slicing
2. Reversing

3. Deleting/Removing a list element
4. Add an item in the list using append and insert method
5. Iterating a list using for loop
6. Replacing a list value with another value
7. Extend a list by another list
8. Find the difference between del and clear operation
9. Sorting the list elements
10. Use of pop in list
11. Reversing the order of the list
12. Finding the max/min value of the list elements
13. Write a program to find the sum of the element in the list
14. Write the program to find the common elements between lists

### **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.

#### **Basic Tuple Operations**

1. Slicing using indexing
2. Finding the max/min value of the list elements
3. Iterating a tuple using for loop
4. Deleting/Removing a tuple element/complete tuple
5. Add an item in the tuple by converting into a list
6. Find the length of the tuple
7. Finding the max/min value of the tuple elements

### **Set**

Python Set is the unordered collection of the data type. It is iterable, mutable (can modify after creation), and has unique elements. In set, the order of the elements is undefined; it may return the changed sequence of the element. The set is created by using a built-in function set (), or a sequence of elements is passed in the curly braces and separated by the comma. It can contain various types of values.

#### **Basic Set Operations**

1. Creating a set
2. looping through the set elements
3. Find the difference between add and update methods
4. Removing items from the set using remove/pop
5. Difference between discard() and remove()
6. Union/Intersection of two Sets

### **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 {}.

#### **Basic Dictionary Operations**

1. Creating and printing a dictionary
2. Accessing Items of a dictionary
3. Use of keys/values and items in dictionary
4. Updating the dictionary
5. Adding/ removing value
6. Use of del/clear functions in dictionary
7. Looping through dictionary keys/values/items

**Code:**

```

# string
s1 = "experiment one two"
s2 = "three"
print(s1+s2)
print(s1*3)
print((s1+s2)[4:10])
print(s1.capitalize()+s2.upper())
print("length of string 1:",len(s1))
print(s1[::-1])

```

```

experiment one twothree
experiment one twoexperiment one two
riment
Experiment one twoTHREE
length of string 1: 18
owt eno tnemirepxe
experimentonetwo

```

```

print(s1.replace(" ", ""))
#lists
list1 = [4,7,20,67,82,1]
list2 = [45,82,9,23]
list1.sort(reverse=True)
print("sorted list:")
print(list1)
print(list1[2:])
print(max(list1),min(list1))
print(list1.remove(7))
list2.extend([12,34,77])
print(list2)
list1.pop(1)

```

```
list1.clear()
print(list1)
del(list1)
print(list1)
```

```
sorted list:
[82, 67, 20, 7, 4, 1]
[20, 7, 4, 1]
82 1
None
[45, 82, 9, 23, 12, 34, 77]
[]
```

Traceback (most recent call last):

```
File "/Users/soham/PycharmProjects/PythonLab/lab 3/e
    print(list1)
NameError: name 'list1' is not defined
```

```
# tuple
```

```
tuple1 = (1,4,87,83,34,12)
tuple2 = (23,56,87,94,13)
print(len(tuple1))
print(tuple1[2:4])
print(max(tuple1))
print(min(tuple2))
sum = 0
for x in tuple2:sum+=x
print(sum)
list1 = list(tuple1)
list1.append(4)
```

```
print(list1)
```

```
6
(87, 83)
87
13
273
[1, 4, 87, 83, 34, 12, 4]
```

```
# dictionary
```

```
d1 = {1:"soham chavan",2:"vikrant",3:"vishal"}
d2 = {"a":"apple" , "b":"banana","c":"cherries"}
print("keys",d1.keys())
print("values",d2.values())
print(d1.items())
d3 = d1.copy()
print(d3)
print(d3.update(d2))
```

```
keys dict_keys([1, 2, 3])
values dict_values(['apple', 'banana', 'cherries'])
dict_items([(1, 'soham chavan'), (2, 'vikrant'), (3, 'vishal')])
{1: 'soham chavan', 2: 'vikrant', 3: 'vishal'}
None
```

```
# sets
```

```
set1 = {1,3,5,7,9}
set2 = {2,4,6,8}
set3 = {1,2,3,4,5,6,7,8,9}
print(set3-set2)
print(set2&set1)
```

```
print(set1|set3)
```

```
print(set1^set2)
```

```
{1, 3, 5, 7, 9}
```

```
set()
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

**Conclusion:**

**Experiment No: 2**

**Aim:** Write python program using control statements

- a. To find out smallest number among three user entered numbers
- b. To print all even numbers in a list
- c. To print all positive numbers in a list
- d. To find Factorial of the given number entered through key board
- e. To count the number of alphabets and digits in a key board entered string
- f. To determine the youngest of three if their ages are entered through key board

**Software Used:** Python

**Theory:**

A programmer may wish to repeat a group of statements several times or he may want to jump from one statement to another statement in the program. For this purpose, we need control statements

**Use of if Statement:**

'if' statement is used to execute one or more statements depending on whether the condition is true or not. The syntax is

```
if condition:
    statements
```

first the condition is tested. If the condition is true, then the statements after: are executed. If the condition is false, the conditions are not executed

**'if ... else' statement**

This statement will execute a group of statements when a condition is true. Otherwise it will execute another group of statements.

**Use of 'for' loop**

'for' loop is used to iterate over the elements of a sequence. It can work with strings, lists, tuple and range etc. The first element of the sequence is assigned to the variable written after for and then the statements are executed. Next the second element is assigned and then the statements are executed. In this way, for each element of the sequence, the statements are executed once. In Python it is possible to use 'else' statement along with for loop or while loop

**Procedure:**

- a. To find out smallest number in a list
  - i. Accept three numbers from users
  - ii. Using if, elif and else statements , identify the smallest number
- b. To print all even numbers in a list
  - i. Accept a list of numbers from the user.
  - ii. Find out the remainder of each list item while dividing by 2
  - iii. If the remainder is '0', the list element represents an even number and if the remainder is '1', it represents an odd number.
  - iv. Implement the above concept using if .. else statement and find out the list of even numbers
- c. To print all positive numbers in a list

- i. Accept a list of numbers from the user.
  - ii. Check whether the given number is less than zero or greater than zero.
  - iii. If the number is greater than 0, identify the given number as a positive number
  - iv. Implement the above concept using if .. else statement and find out the list of even numbers
- d. To find Factorial of the given number entered through
  - i. Accept the number from the user
  - ii. Using for loop try to find out the factorial of the given number using the concept of  $n! = n \times (n-1)!$
- e. To count the number of alphabets and digits in a key board entered string
  - a. Accept a string with alphabets and digits
  - b. Iterate through the complete string using for loop and then using 'isalpha' function with if control statement count the number of alphabets present in the string and similarly using 'isdigit' function, find the number of digits present in the given string
- f. To determine the youngest of three if their ages are entered through key board
  - i. Accept three ages from users
  - ii. Using if, elif and else statements , identify the youngest
  - iii. Print the name of the youngest child

**Code:**

```
l1 = [1,3, 6,-2,-6,9,39,-24]

#smallest number
print(l1.min())

#even numbers
for x in l1:
    if(x%2==0):
        print(x)

#positive number
for x in l1:
    if(x == abs(x)):
        print(x)
```



Output:

-24

6

-2

-6

-24

1

3

6

9

39

Conclusion:

### Experiment No: 3

**Aim:** Write a program to

- a. to receive two integers from key board and get their sum , product, subtraction through a user defined function
- b. To define a function which computes the frequency of words present in a string passed to it.
- c. To define a function to calculate the gross salary of an employee where gross salary is defined as basic salary+ DA+ HRA. DA is 80% of basic salary and HRA is 20% of basic salary. User input is basic salary.
- d. To define function to accept a group of numbers from keyboard and find their total and average

**Software Used:** Python

#### Theory:

Functions are the most important aspect of an application. A function can be defined as the organized block of reusable code, which can be called whenever required. Python allows us to divide a large program into the basic building blocks known as a function. A function can be called multiple times to provide reusability. There are mainly two types of functions.

User-define functions: The user-defined functions are those define by the user to perform the specific task.

Built-in functions: The built-in functions are those functions that are pre-defined in Python

Function blocks begin with the keyword 'def' followed by the function name and parentheses In Python, after the function is created, we can call it from another function. A function must be defined before the function call; otherwise, the Python interpreter gives an error. To call the function, use the function name followed by the parentheses. The return statement is used at the end of the function and returns the result of the function. It terminates the function execution and transfers the result where the function is called. The return statement cannot be used outside of the function. The arguments are types of information which can be passed into the function. The arguments are specified in the parentheses. We can pass any number of arguments, but they must be separate them with a comma. There may be several types of arguments which can be passed at the time of function call. The Python allows us to provide the mix of the required arguments and keyword arguments at the time of function call. However, the required argument must not be given after the keyword argument, i.e., once the keyword argument is encountered in the function call, the following arguments must also be the keyword arguments

#### Code:

```
# Factorial

n = int(input("enter number for factorial "))

f=1

for x in range(1,n+1):
```

```

f = f*x
print("factorial = ",f)

```

```

/0ser3/sonam/pythonml-projects/pythonlab/venv/
enter number for factorial 12
factorial = 479001600

```

*# count of  
characters in a  
string*

```

s = input("enter string : ")
alpha = 0
num = 0
for x in s:
    if(x.isalpha()):
        alpha+=1
    if(x.isdigit()):
        num+=1
print("characters : ",alpha)
print("number : ",num)
# Youngest element

```

```

enter string : Lc2q9w70uqcihbzcbY8ebc29eyq9ce20euxb2he xehn28ebce2bc
characters : 38
number : 14

```

```

l=list(map(int,input("enter ages: ").split()))
print(" youngest age is :",min(l))

```

```
enter ages: 12 35 81 62 91 3 01 27  
youngest age is : 1
```

**Conclusion:**

**Experiment No: 4**

**Aim:** Write a program to

- e. to receive two integers from key board and get their sum , product, subtraction through a user defined function
- f. To define a function which computes the frequency of words present in a string passed to it.
- g. To define a function to calculate the gross salary of an employee where gross salary is defined as basic salary+ DA+ HRA. DA is 80% of basic salary and HRA is 20% of basic salary. User input is basic salary.
- h. To define function to accept a group of numbers from keyboard and find their total and average

**Software Used:** Python

**Theory:**

Functions are the most important aspect of an application. A function can be defined as the organized block of reusable code, which can be called whenever required. Python allows us to divide a large program into the basic building blocks known as a function. A function can be called multiple times to provide reusability. There are mainly two types of functions.

User-define functions: The user-defined functions are those define by the user to perform the specific task.

Built-in functions: The built-in functions are those functions that are pre-defined in Python

Function blocks begin with the keyword 'def' followed by the function name and parentheses In Python, after the function is created, we can call it from another function. A function must be defined before the function call; otherwise, the Python interpreter gives an error. To call the function, use the function name followed by the parentheses. The return statement is used at the end of the function and returns the result of the function. It terminates the function execution and transfers the result where the function is called. The return statement cannot be used outside of the function. The arguments are types of information which can be passed into the function. The arguments are specified in the parentheses. We can pass any number of arguments, but they must be separate them with a comma. There may be several types of arguments which can be passed at the time of function call. The Python allows us to provide the mix of the required arguments and keyword arguments at the time of function call. However, the required argument must not be given after the keyword argument, i.e., once the keyword argument is encountered in the function call, the following arguments must also be the keyword arguments

**Code:**

```
# math function

def simp_math(a,b):

    print(a+b)

    print(a-b)
```

```

    print(a/b)

    print(a*b)

num1 = int(input(print("enter value 1: ")))
num2 = int(input(print("enter value 2: ")))
simp_math(num1,num2)

```

```

enter value 1:
None23
enter value 2:
None56
79
-33
0.4107142857142857
1288

```

```

# frequency function

```

```

def freq(str):

    str = str.split()

```

```

    str2 = []

    for i in str:

        if i not in str2:

            # insert value in str2

            str2.append(i)

    for i in range(0, len(str2)):

        print('Frequency of', str2[i], 'is :', str.count(str2[i]))

def main():

    str = input(print("enter string"))

    freq(str)

if __name__ == "__main__":

```

```
main()
```

```
Noneapple mango apple orange orange apple guava mango mango
Frequency of apple is : 3
Frequency of mango is : 3
Frequency of orange is : 2
Frequency of guava is : 1
```

```
# gross salary function
```

```
def grossSalary(sal,DA,HRA):
```

```
    gsal=0.0
```

```
    DA = sal*(DA/100)
```

```
    HRA = sal*(HRA/100)
```

```
    gsal = sal+DA+HRA
```

```
    return gsal
```

```
sal = float(input(print(" enter salary: ")))
```

```
da = float(input(print(" enter da: ")))
```

```
hda = float(input(print(" enter hda: ")))
```

```
print("gross salary =
```

```
enter salary:
None100000
enter da:
None5
enter hda:
None7
gross salary = 112000.000000
```

```
%f"%(grossSalary(sal,da,hda))
```

*# largest number function*

**def** funCTion(l):

    sum = 0

**for** x **in** l:

        sum+=x

    length = len(l)

    print(" average = %f"%(sum/length))

    print(" sum = %f" % (sum ))

l = list(map(int,input("enter elements: ").split()))

funCTion(l)

```
enter elements: 12 82 03 658 92 92 84 29
average = 131.500000
sum = 1052.000000
```

**CONCLUSION:**



**Experiment No 5:**

**Aim:** Write a program to do the following input and output operations on the given text file.

- a. read
- b. write
- c. open
- d. close
- e. append
- f. readlines
- g. for loop for reading lines
- h. tell
- i. seek

**Software Used:** Python

**Theory**

Opening Files in Python: Python has a built-in `open()` function to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly. To create a new file in Python, use the `open()` method, with one of the following parameters:

- "x" - Create - will create a file, returns an error if the file exist
- "a" - Append - will create a file if the specified file does not exist
- "w" - Write - will create a file if the specified file does not exist

We can specify the mode while opening a file. In mode, we specify whether we want to read 'r', write 'w' or append 'a' to the file. We can also specify if we want to open the file in text mode or binary mode. The default is reading in text mode. In this mode, we get strings when reading from the file. On the other hand, binary mode returns bytes and this is the mode to be used when dealing with non-text files like images or executable files. We need to be careful with the w mode, as it will overwrite into the file if it already exists. Due to this, all the previous data are erased

**Code:**

```
import os

f = open("file_1.txt", 'r+')

print(f.read())

f.write("\n"+input(" add text "))

print(f.tell())

f.seek(0)

print(f.read())

print(f.tell())

f.write("\n"+input(" add text2 "))

print(f.tell())

f.seek(0)
```

```

print(f.read())

os.rename("file_1.txt", "myfile.txt")

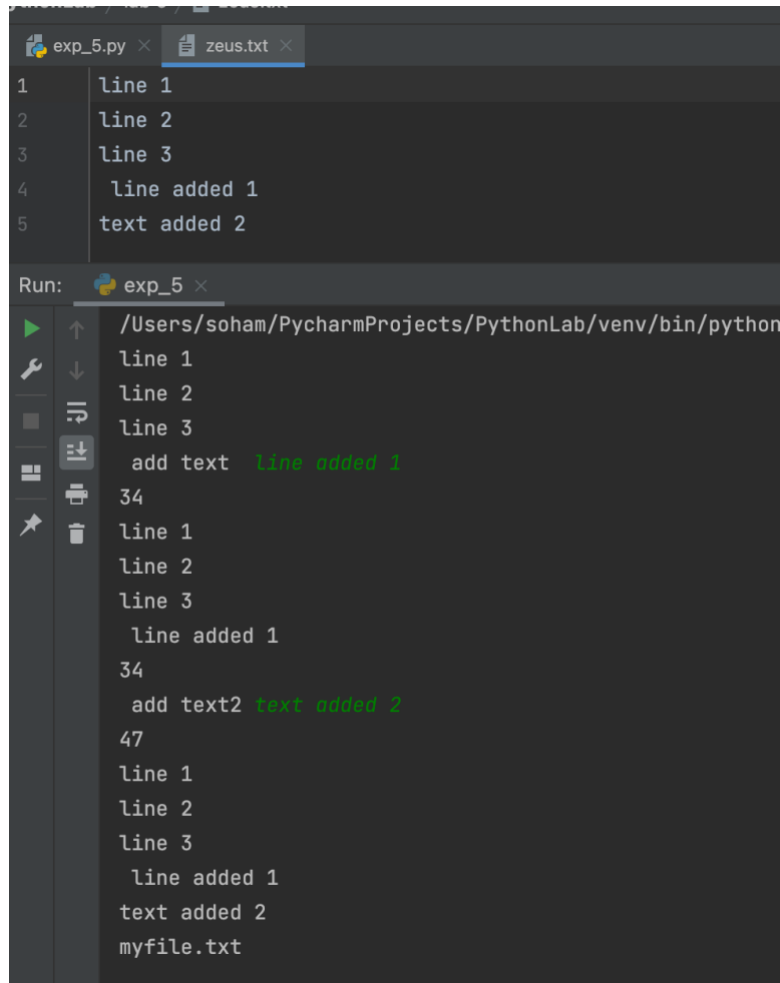
print(f.name)

for x in range(0,3):

    f.readline(x)

```

**Output:**



The screenshot shows the PyCharm IDE interface. At the top, there are two tabs: 'exp\_5.py' and 'zeus.txt'. The 'zeus.txt' tab is active, displaying the following text:

```

1 line 1
2 line 2
3 line 3
4 line added 1
5 text added 2

```

Below the editor, the 'Run' window is open, showing the execution of 'exp\_5.py'. The output is as follows:

```

/Users/soham/PycharmProjects/PythonLab/venv/bin/python
line 1
line 2
line 3
add text line added 1
34
line 1
line 2
line 3
line added 1
34
add text2 text added 2
47
line 1
line 2
line 3
line added 1
text added 2
myfile.txt

```

**Conclusion:**

## **Experiment No 6:**

**Aim:** Write a program to

- a. prepare a class of students with name, age, Id. Arrange it as per name and display all the details
- b. to understand the multiple inheritance concept. Include the concept of constructor overriding

**Software Used:** Python

### **Theory**

An object is simply a collection of data (variables) and methods (functions) that act on those data. Similarly, a class is a blueprint for that object. We can think of a class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions we build the house. House is the object. A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by their class) for modifying their state.

- 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 (.)

When an object of a class is created, the class is said to be instantiated. All the instances share the attributes and the behaviour of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any number of instances. Class methods must have an extra first parameter 'self' in the method definition. We do not give a value for this parameter when we call the method, Python provides it.

### **Defining a Class in Python**

Like function definitions begin with the 'def' keyword in Python, class definitions begin with a 'class' keyword. The first string inside the class is called 'docstring' and has a brief description of the class. Although not mandatory, this is highly recommended. As soon as we define a class, a new class object is created with the same name. This class object allows us to access the different attributes as well as to instantiate new objects of that class.

### **Constructors in Python**

Class functions that begin with double underscore \_\_ are called special functions as they have special meaning. Of one particular interest is the \_\_init\_\_ () function. This special function gets called whenever a new object of that class is instantiated. This type of function is also called constructors in Object Oriented Programming (OOP). We normally use it to initialize all the variables. We can pass any number of arguments at the time of creating the class object, depending upon the \_\_init\_\_ () definition. It is mostly used to initialize the class attributes. Every class must have a constructor, even if it simply relies on the default constructor.

### **Python Inheritance**

Inheritance is an important aspect of the object-oriented paradigm. Inheritance provides code reusability to the program because we can use an existing class to create a new class instead of creating it from scratch. In inheritance, the

child class acquires the properties and can access all the data members and functions defined in the parent class. A child class can also provide its specific implementation to the functions of the parent class. In python, a derived class can inherit base class by just mentioning the base in the bracket after the derived class name. Multi-Level inheritance is possible in python like other object-oriented languages. Multi-level inheritance is achieved when a derived class inherits another derived class. There is no limit on the number of levels up to which, the multi-level inheritance is achieved in python.

#### **Creating the constructor in python**

In Python, the method the `__init__()` simulates the constructor of the class. This method is called when the class is instantiated. It accepts the self-keyword as a first argument which allows accessing the attributes or method of the class.

**Method Overriding:** We can provide some specific implementation of the parent class method in our child class. When the parent class method is defined in the child class with some specific implementation, then the concept is called method overriding. We may need to perform method overriding in the scenario where the different definition of a parent class method is needed in the child class.

**Python constructor overriding:** It means one method will override the other. The parent class and child class both have the constructor and the child will override the parent constructor. When you override the constructor, the constructor from the parent class that we inherited is not called at all. If you still want that functionality, you have to call it yourself. This is done with `super()`: it returns a reference to the parent class, so we can call the parent class's constructor.

#### **Code:**

```
# Class Student

class student():

    age = 0

    name = ""

    id = 0

    def SpillParameters(self):

        print(self.name)

        print(self.id)

        print(self.age)
```

```

obj1 = student()
obj1.age = int(input(print(" onj 1 age"))))
obj1.id = int(input(print(" onj 1 id"))))
obj1.name = input(print(" onj 1 name"))

obj2 = student()
obj2.age = int(input(print(" onj 2 age"))))
obj2.id = int(input(print(" onj 2 id"))))
obj2.name = input(print(" onj 2 name"))

obj1.SpillParameters()
obj2.SpillParameters()

```

```

1 age
None12
1 id
None34
1 name
Noneadam levine
2 age
None23
2 id
None56
2 name
Nonejohn legend
adam levine
34
12
john legend
56
23

```

```

# mltiple inheritance

class TeamMember(object):

    def __init__(self, name, uid):

        self.name = name

        self.uid = uid

class Worker(object):

```

```
def __init__(self, pay, jobtitle):
    self.pay = pay
    self.jobtitle = jobtitle

class TeamLeader(TeamMember, Worker):
    def __init__(self, name, uid, pay, jobtitle, exp):
        self.exp = exp
        TeamMember.__init__(self, name, uid)
        Worker.__init__(self, pay, jobtitle)
        print("Name: {}, Pay: {}, Exp: {}".format(self.name, self.pay,
self.exp))

TL = TeamLeader('Jake', 10001, 250000, 'Scrum Master', 5)
```

```
Name: Jake, Pay: 250000, Exp: 5
```

**Conclusion:**

## Experiment No 7:

**Aim:** Write a program to

- a. Create a database with name my database and add table with name students. Add the Roll no name, age, mobile no details of these students using insert operation. Update the database for one more student. Display the database entries. Delete one entry in the database
- b. Write Python program to study define, edit arrays and perform arithmetic operations using numpy.
- c. Write python program to study selection, indexing, merging, joining, and concatenation in data frames

**Software Used:** Python with SQLite library

### Theory:

SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. SQLite is a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day. SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. To use the module, you must first create a Connection object that represents the database.

### SQLite Commands

The standard SQLite commands to interact with relational databases are similar to SQL. They are CREATE, SELECT, INSERT, UPDATE, DELETE and Replace.

- a. Select - query data from a single table using SELECT statement
- b. Insert - insert rows into a table
- c. Update - update existing rows in a table.
- d. Delete - delete rows from a table.
- e. Replace - insert a new row or replace the existing row in a table.
- f. Create Table - show you how to create a new table in the database.

### Code:

```
#database

import sqlite3 as sql

base = sql.connect("mydatabase.db")

base.execute('insert into Students values( 2,"sam smith",34,1234567890)');
```

```

base.execute('insert into Students values(?,?,?,?)',(int(input(print("Roll
no"))),input(print("Name")),int(input(print("Age"))),int(input(print("Mobile
no")))))

base.commit()

cursor = base.execute("SELECT Roll No, Name, Age, Mobile No from Students")

for row in cursor:

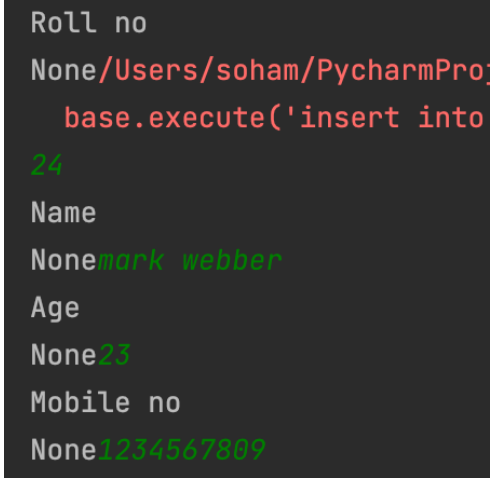
    print ("Roll No = ", row[0])

    print ("NAME = ", row[1])

    print ("AGE = ", row[2])

                                print ("MOBILE NO = ", row[3], "\n")

```



```

Roll no
None/24
Name
None/mark webber
Age
None/23
Mobile no
None/1234567809

```

```

#arrays

import numpy as np

arr1 = np.array([[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15]])

arr2 = np.array([[2,2,2,2,2],[3,3,3,3,3],[4,4,4,4,4]])

print("array1 +3 = \n ",arr1 + 3)

print(" array1 + array2 =")

print(arr1 + arr2)

arr3 = np.vstack((arr1,arr2))

print('vertically stacked array ->')

print(arr3)

```



```
arr3 = np.hstack((arr1,arr2))

print('horizontally stacked array ->')

print(arr3)
```

```
array1 +3 =
[[ 4  5  6  7  8]
 [ 9 10 11 12 13]
 [14 15 16 17 18]]
array1 + array2 =
[[ 3  4  5  6  7]
 [ 9 10 11 12 13]
 [15 16 17 18 19]]
vertically stacked array ->
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [ 2  2  2  2  2]
 [ 3  3  3  3  3]
 [ 4  4  4  4  4]]
horizontally stacked array ->
[[ 1  2  3  4  5  2  2  2  2  2]
 [ 6  7  8  9 10  3  3  3  3  3]
 [11 12 13 14 15  4  4  4  4  4]]
[[ 1.55740772e+00 -2.18503986e+00 -1.42546543e-01  1.15782128e+00
 -3.38051501e+00]
 [-2.91006191e-01  8.71447983e-01 -6.79971146e+00 -4.52315659e-01
  6.48360827e-01]
 [-2.25950846e+02 -6.35859929e-01  4.63021133e-01  7.24460662e+00
 -8.55993401e-01]]
```

```
print(np.tan(arr1))
```

**Conclusion:**

**Experiment No 8:**

**Aim:** Write a program using SciPy to

- a. Solve a linear algebra problem. There is a test with 40 questions worth 200 marks. The test has two types of questions: 1. true or false - carries 4 marks each      2. Multiple-choice - carries 9 marks each. Find the number of true or false and multiple-choice questions. Also
- b. find the Eigen values and Eigen vectors of the given matrix
- c. Verify FFT and IFFT operations on a numpy array

**Software used:** Python with SciPy Library

**Theory:**

SciPy provides algorithms for optimization, integration, interpolation, eigenvalue problems, algebraic equations, differential equations, statistics and many other classes of problems. SciPy is a scientific computation library that uses NumPy underneath. SciPy stands for Scientific Python. It provides more utility functions for optimization, stats and signal processing. Like NumPy, SciPy is open source so we can use it freely.

- SciPy contains varieties of sub packages which help to solve the most common issue related to Scientific Computation.
- SciPy package in Python is the most used Scientific library only second to GNU Scientific Library for C/C++ or Matlab's.
- Easy to use and understand as well as fast computational power.
- It can operate on an array of NumPy library.

**Linear Algebra with SciPy**

Linear algebra routine accepts two-dimensional array object and output is also a two-dimensional array. It is implemented with the help of `scipy.linalg` module.

**Linear Equations**

The `scipy.linalg.solve` feature solves the linear equation  $a * x + b * y = Z$ , for the unknown  $x, y$  values. The solve function takes two inputs 'a' and 'b' in which 'a' represents the coefficients and 'b' represents the respective right hand side value and returns the solution array.

**Eigenvalues and Eigenvector-`scipy.linalg.eig ()`**

The most common problem in linear algebra is eigenvalues and eigenvector which can be easily solved using `eig ()` function. We can find the Eigenvalue of (X) and correspond eigenvector of a two-dimensional square matrix. The eigenvalue-eigenvector problem is one of the most commonly employed linear algebra operations. We can find the Eigen values ( $\lambda$ ) and the corresponding Eigen vectors ( $v$ ) of a square matrix (A) by considering the following relation -  $Av = \lambda v$ . `scipy.linalg.eig` computes the eigenvalues from an ordinary or generalized eigenvalue problem. This function returns the Eigen values and the Eigen vectors.

### Discrete Fourier Transform - scipy.fftpack

DFT is a mathematical technique which is used in converting spatial data into frequency data.

FFT (Fast Fourier Transformation) is an algorithm for computing DFT. FFT is applied to a multidimensional array. Frequency defines the number of signal or wavelength in particular time period.

Code:

```
from scipy import linalg
import numpy as np

# The function takes two arrays
a = np.array([[1, 1], [4, 9]])      #  $x + y = 40$  ,  $4x + 9y = 200$ 
b = np.array([40, 200])

# Solving the linear equations
res = linalg.solve(a, b)

print("no of true and false questions is %d and \n multiple choice questions  
are %d"%(res[0],res[1]))
```

```
no of true and false questions is 32 and
multiple choice questions are 8
```

```
import numpy as np

# create numpy 2d-array
m = np.array([[1, 13, 2],
              [2, 7, 4],
              [4, 9, 6]])

print("Original 2D array:\n",m)

w, v = np.linalg.eig(m)

# printing eigen values
print("Eigen values:\n",w)

# printing eigen vectors
```

```
print("eigenvectors:\n",v)
```

```
Original 2D array:
[[ 1 13  2]
 [ 2  7  4]
 [ 4  9  6]]
Eigen values:
[15.15520626+0.j          -0.57760313+1.47436895j -0.57760313-1.47436895j]
eigenvectors:
[[ 0.53438555+0.j          0.85898739+0.j          0.85898739-0.j          ]
 [ 0.47423067+0.j          -0.04258777+0.14090016j -0.04258777-0.14090016j]
 [ 0.69966946+0.j          -0.4007501 -0.2826189j  -0.4007501 +0.2826189j  ]]
```

```
from sympy import fft
import numpy as np
seq = np.array([14, 21, 13, 44])
transform = fft(seq)

print(transform)
```

```
[92, 1 - 23*I, -38, 1 + 23*I]
```

**Conclusion:**

**Experiment No 9:**

**Aim:** Write a program using OnenCV to

- a. Get gray scale image from colour image
- b. get the histogram of the given image
- c. get low pass filtering of the given image
- d. get the histogram equalization of the given image
- e. to get edge detection of the given image
- f. Write a python program to find image morphological operations of the given image

**Software used:** Python with OpenCV Library

**Theory:**

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. 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.

**Image Processing**

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. Grayscale images are those images which contain only two colours 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. OpenCV allows us to perform multiple operations on the image, but to do that it is necessary to read an image file as input, and then we can perform the various operations on it.

**Image Reading/ writing/showing**

OpenCV provides following functions which are used to read and write the images. The imread () function loads image from the specified file and returns it. OpenCV imwrite () function is used to save an image to a specified file. Cv2.imshow function is used to display the image on the screen

**RGB to Grayscale conversion**

The cvtColor is used to convert an image from one colour space to another. Using cv2.COLOR\_BGR2GRAY is used for converting BGR image to grayscale. Edge detection is term where identify the boundary of object in image. We will learn about the edge detection using the canny edge detection technique.

**Image Blurring/Averaging**

Blurring is the commonly used technique for image processing to removing the noise. It is generally used to eliminate the high-frequency content such as noise, edges in the image. The edges are being blurred when we apply blur to the image. OpenCV provides mainly the following type of blurring techniques.

In averaging technique, the image is convolved with a box filter (normalize). It calculates the average of all the pixels which are under the kernel area and replaces the central element with the calculated average. OpenCV provides the `cv2.blur ()` function for blurring using averaging

The median blur operation is quite similar to the Gaussian blur. OpenCV provides the `medianblur ()` function to perform the blur operation. It takes the median of all the pixels under the kernel area, and the central element is replaced with this median value. It is extremely effective for the salt-and-paper noise in the image.

### **OpenCV Erosion and Dilation**

Erosion and Dilation are morphological image processing operations. OpenCV morphological image processing is a procedure for modifying the geometric structure in the image. In morphism, we find the shape and size or structure of an object. Both operations are defined for binary images, but we can also use them on a grayscale image. These are widely used in the following way:

- Removing Noise
- Identify intensity bumps or holes in the picture.
- Isolation of individual elements and joining disparate elements in image.

In this tutorial, we will explain the erosion and dilation briefly.

### **Dilation**

Dilation is a technique where we expand the image. It adds the number of pixels to the boundaries of objects in an image. The structuring element controls it. The structuring element is a matrix of 1's and 0's.

### **Structuring Element**

The size and shape of the structuring element define how many numbers of the pixel should be added or removed from the objects in an image. It is a matrix of 1's and 0's. The centre pixel of the image is called the origin. It contains an image A with some kernel B. The dilation operation is performed by using the `cv2.dilate ()` method.

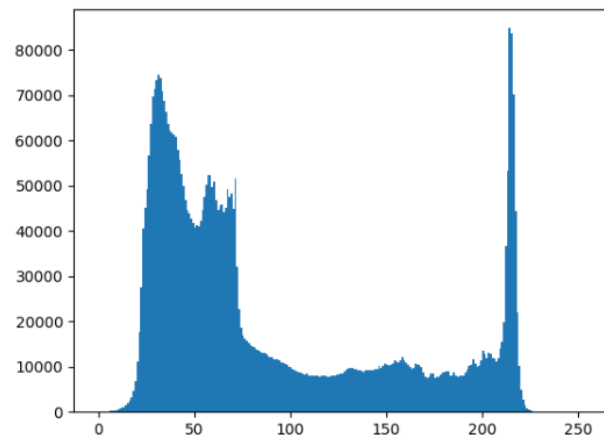
### **Erosion**

Erosion is much similar to dilation. The difference is that the pixel value calculated minimum rather than the maximum in dilation. The image is replaced under the anchor point with that calculated minimum pixel. Unlike dilation, the regions of darker shades increase. While it decreases in white shade or brighter side. OpenCV provides `cv2.erode ()` function to perform this operation.

### **Image Histogram**

An image histogram is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value. Histogram is a graphical representation of the intensity distribution of an image. Histogram quantifies the number of pixels for each intensity value. As a common terminology, each possible gray value is referred to as a bin, and the count is referred to as frequency. In histogram equalization, we are interested in the intensity histogram of the image. That means for each possible pixel intensity value (0 to 255), we count the number of pixels having the corresponding value. Histogram is considered as a graph or plot which is related to frequency of pixels in a Gray Scale Image with pixel values (ranging from 0 to 255). Grayscale image is an image in which the

value of each pixel is a single sample, that is, it carries only intensity information where pixel value varies from 0 to 255. To create a histogram of our image data, we use the `hist ()` function.



### Histogram Equalization

Histogram equalization is a basic image processing technique that adjusts the global contrast of an image by updating the image histogram's pixel intensity distribution. The result of applying histogram equalization is an image with higher global contrast. The most widely used histogram equalization application can be found in the medical field. Histogram equalization is a simple technique to enhance a digital image. It is a standard tool in digital image processing and computer vision applications. It is especially effective in improving the visual quality of grayscale images. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values. The method is useful in images with backgrounds and foregrounds that are both bright or both dark. OpenCV has a function to do this, `cv2.equalizeHist ()`. Its input is just grayscale image and output is our histogram equalized image. Histogram equalization has been applied extensively in medical imaging to improve the contrast of X-ray and MRI images. Such improvement enables more accurate medical diagnosis.

Code:

```
import matplotlib.pyplot as plt

import numpy as np

import cv2

img = cv2.imread('barrelImage.jpg')

R,G,B = cv2.split(img)

cv2.imshow('Image Original',img)
```

```

req = cv2.equalizeHist(R)
geq = cv2.equalizeHist(G)
beq = cv2.equalizeHist(B)
imeq = cv2.merge((req,geq,beq))

np.vstack((plt.hist(req.ravel(),256,[0,256]),plt.hist(geq.ravel(),256,[0,256]
),plt.hist(beq.ravel(),256,[0,256])))

plt.show()

edges = cv2.Canny(img,60,200)

cv2.imshow('edges of image',edges)

img2 = cv2.imread("AppleTvLogo.png")

kernel = np.ones((5,5),np.uint8)

dilation = cv2.dilate(img2,kernel,iterations = 3)

erode = cv2.erode(img2,kernel,iterations = 3)

cv2.imshow("orioginal",img2)

cv2.imshow("dial",dilation)

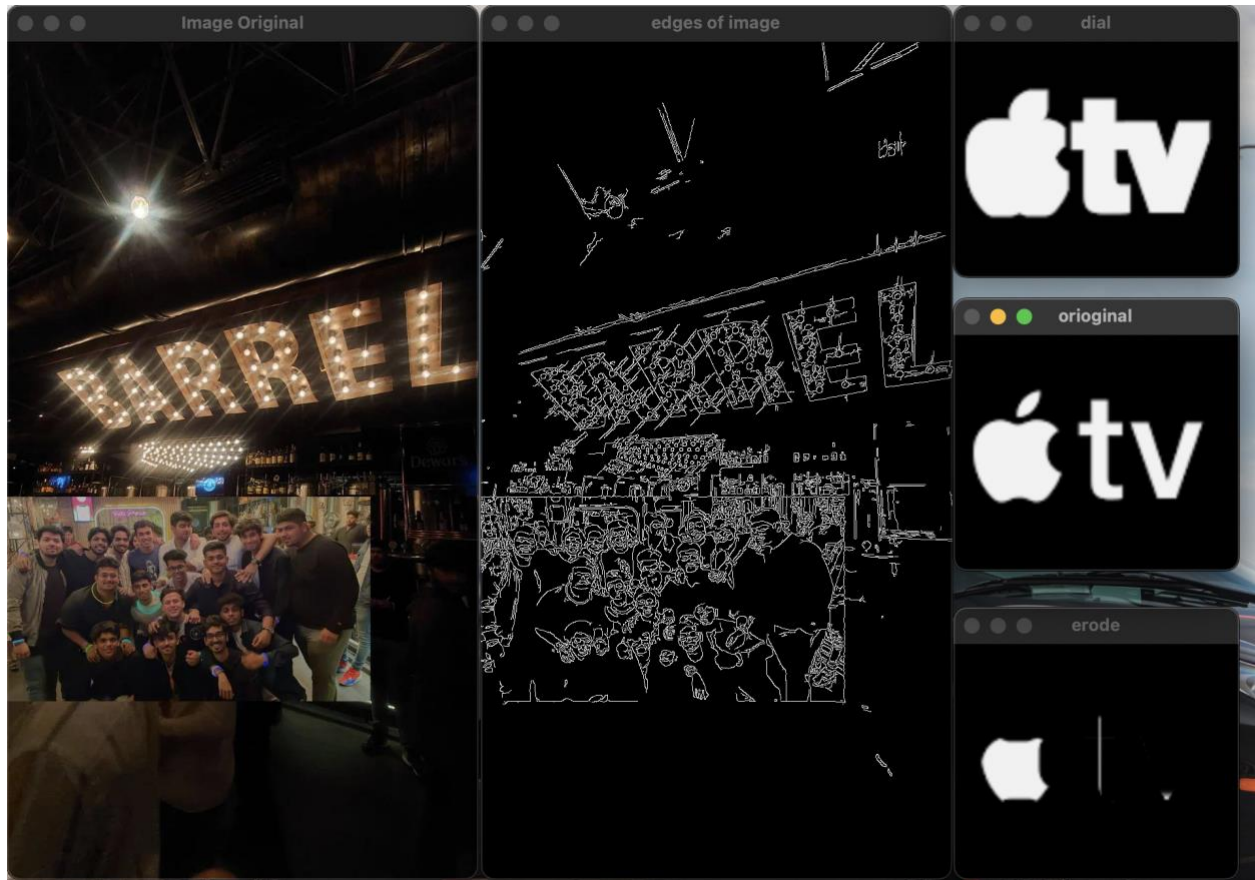
cv2.imshow("erode",erode)

cv2.waitKey(0)

```

**Output:**





Conclusion:

## Experiment No: 10

**Aim:** Write a program using sclera machine learning library to

- a. to study multiple linear regression
- b. to study logistic regression
- c. to study Support Vector Machine

**Software Used:** Python

### Theory:

#### Linear Regression

Regression analysis is one of the most important fields in statistics and machine learning. There are many regression methods available. Linear regression is one of them. Regression searches for relationships among variables. You can try to establish a mathematical dependence of the prices of houses on their areas, numbers of bedrooms, distances to the city centre, and so on using regression. Regression problems usually have one continuous and unbounded dependent variable. The inputs, however, can be continuous, discrete, or even categorical data such as gender, nationality, brand, and so on. It is a common practice to denote the outputs with  $y$  and inputs with  $x$ . If there are two or more independent variables, they can be represented as the vector  $\mathbf{x} = (x_1, \dots, x_r)$ , where  $r$  is the number of inputs.

Linear regression is probably one of the most important and widely used regression techniques. It's among the simplest regression methods. One of its main advantages is the ease of interpreting results. When implementing linear regression of some dependent variable  $y$  on the set of independent variables  $\mathbf{x} = (x_1, \dots, x_r)$ , where  $r$  is the number of predictors, you assume a linear relationship between  $y$  and  $\mathbf{x}$ :  $y = \beta_0 + \beta_1 x_1 + \dots + \beta_r x_r + \varepsilon$ . This equation is the regression equation.  $\beta_0, \beta_1, \dots, \beta_r$  are the regression coefficients, and  $\varepsilon$  is the random error. Linear regression calculates the estimators of the regression coefficients or simply the predicted weights, denoted with  $b_0, b_1, \dots, b_r$ . They define the estimated regression function  $(\mathbf{x}) = b_0 + b_1 x_1 + \dots + b_r x_r$ . This function should capture the dependencies between the inputs and output sufficiently well. The value of  $b_0$ , also called the intercept, shows the point where the estimated regression line crosses the  $y$  axis. It is the value of the estimated response  $(x)$  for  $x = 0$ . The value of  $b_1$  determines the slope of the estimated regression line.

The residuals can be calculated as  $y_i - (\mathbf{x}_i) = y_i - b_0 - b_1 x_i$  for  $i = 1, \dots, n$ .

#### Multiple Linear Regression

Multiple or multivariate linear regression is a case of linear regression with two or more independent variables. If there are just two independent variables, the estimated regression function is  $(x_1, x_2) = b_0 + b_1 x_1 + b_2 x_2$ . It represents a regression plane in a three-dimensional space. The goal of regression is to determine the values of the weights  $b_0, b_1$ , and  $b_2$  such that this plane is as close as possible to the actual responses and yield the minimal SSR.

The case of more than two independent variables is similar, but more general.

The estimated regression function is  $(x_1, \dots, x_r) = b_0 + b_1 x_1 + \dots + b_r x_r$ , and there are  $r + 1$  weights to be determined when the number of inputs is  $r$ .

The package scikit-learn is a widely used Python library for machine learning, built on top of NumPy and some other packages. It provides the means for pre-processing data, reducing dimensionality, implementing regression, classification, clustering, and more. Like NumPy, scikit-learn is also open source. There are five basic steps when you're implementing linear regression:

1. Import the packages and classes you need.
2. Provide data to work with and eventually do appropriate transformations.
3. Create a regression model and fit it with existing data.
4. Check the results of model fitting to know whether the model is satisfactory.
5. Apply the model for predictions.

### **Logistic Regression**

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts  $P(Y=1)$  as a function of  $X$ . Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence. It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

Linear Regression Equation:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

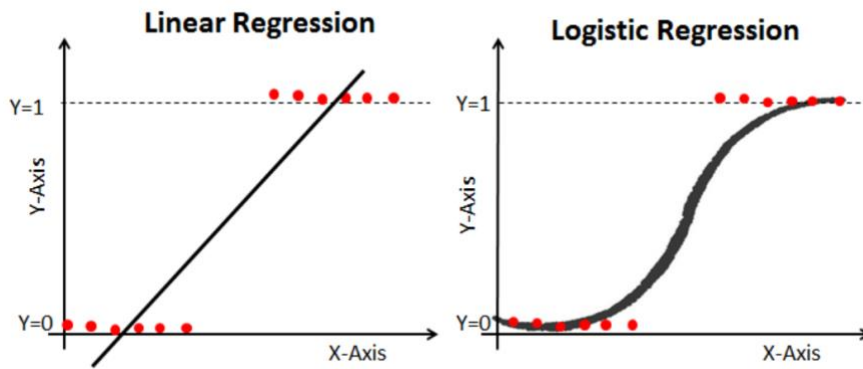
Where,  $y$  is dependent variable and  $x_1, x_2 \dots$  and  $X_n$  are explanatory variables.  
Sigmoid Function:

$$p = 1 / (1 + e^{-y})$$

Apply sigmoid function on linear regression:

$$p = 1 / (1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n)})$$

Linear regression gives you a continuous output, but logistic regression provides a constant output. An example of the continuous output is house price and stock price.



1. Import the packages and classes you need.
2. Provide data to work with and eventually do appropriate transformations.
3. Create a logistic regression model and fit it with existing data.
4. Apply the model for predictions
5. Evaluate the model using Confusion Matrix

### Support Vector machine (SVM)

SVM is a supervised and linear Machine Learning algorithm most commonly used for solving classification problems and is also referred to as Support Vector Classification. There is also a subset of SVM called SVR which stands for Support Vector Regression which uses the same principles to solve regression problems. SVM also supports the kernel method also called the kernel SVM which allows us to tackle non-linearity. SVM generates a line that can cleanly separate the two classes. How clean, you may ask. There are many possible ways of drawing a line that separates the two classes, however, in SVM, it is determined by the margins and the support vectors. The margin is the area separating the two dotted green lines as shown in the image above. The more the margin the better the classes are separated. The support vectors are the data points through which each of the green lines passes through. These points are called support vectors as they contribute to the margins and hence the classifier itself. These support vectors are simply the data points lying closest to the border of either of the classes which has a probability of being in either one.

The SVM then generates a hyper plane which has the maximum margin, in this case the black bold line that separates the two classes which is at an optimum distance between both the classes

```
import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets, linear_model, metrics

# load the boston dataset

boston = datasets.load_boston(return_X_y=False)
```

```

# defining feature matrix(X) and response vector(y)

X = boston.data

y = boston.target

# splitting X and y into training and testing sets

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
random_state=1)

# create linear regression object

reg = linear_model.LinearRegression()

# train the model using the training sets

reg.fit(X_train, y_train)

# regression coefficients

print('Coefficients: ', reg.coef_)

# variance score: 1 means perfect prediction

print('Variance score: {}'.format(reg.score(X_test, y_test)))

# plot for residual error


## setting plot style

plt.style.use('fivethirtyeight')

## plotting residual errors in training data

plt.scatter(reg.predict(X_train), reg.predict(X_train) - y_train,
            color="green", s=10, label='Train data')

## plotting residual errors in test data

plt.scatter(reg.predict(X_test), reg.predict(X_test) - y_test,
            color="blue", s=10, label='Test data')

## plotting line for zero residual error

plt.hlines(y=0, xmin=0, xmax=50, linewidth=2)

## plotting legend

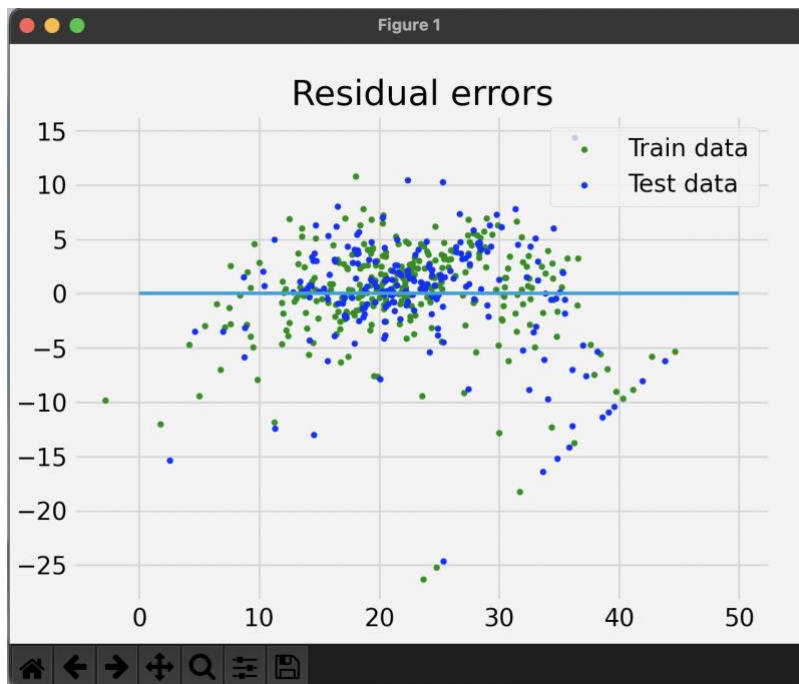
plt.legend(loc='upper right')

## plot title

```

```
plt.title("Residual errors")  
  
## method call for showing the plot  
  
plt.show()
```

```
Coefficients: [-8.95714048e-02  6.73132853e-02  5.04649248e-02  2.18579583e+00  
-1.72053975e+01  3.63606995e+00  2.05579939e-03 -1.36602886e+00  
 2.89576718e-01 -1.22700072e-02 -8.34881849e-01  9.40360790e-03  
-5.04008320e-01]  
Variance score: 0.7209056672661762
```



## Python Mini Project

**Aim:** to predict house taxes In Boston with given data.

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.metrics import r2_score

data = pd.read_csv('HousingData.csv')

data.head()

data.describe()

data.info()

data.corr()['TAX'].sort_values(ascending=False)

col = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'AGE', 'LSTAT']

for c in col:

    data[c].fillna(data[c].mean(), inplace=True)

data.info()

from sklearn.preprocessing import StandardScaler

X = data.drop('TAX', axis=1)

y = data['TAX']

scaler = StandardScaler()

X = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

from sklearn.ensemble import RandomForestRegressor

from sklearn.model_selection import GridSearchCV

rfc = RandomForestRegressor()

params = {'n_estimators':[100,200,300,400,500,600,700,800,900,1000]}

grid_model = GridSearchCV(rfc, params, verbose=2)

grid_model.fit(X_train,y_train)

pred = grid_model.predict(X_test)
```

```

print('Random Forest accuracy is --> ', r2_score(y_test, pred)*100)

grid_model.best_params_

res = pd.DataFrame()

res['Y_Test'] = y_test

res['PRED'] = pred

res.head()

sns.scatterplot(y_test, pred)

plt.xlabel('real values')

plt.ylabel('predicted values')

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  -
0   CRIM        486 non-null    float64
1   ZN          486 non-null    float64
2   INDUS       486 non-null    float64
3   CHAS        486 non-null    float64
4   NOX         506 non-null    float64
5   RM          506 non-null    float64
6   AGE         486 non-null    float64
7   DIS         506 non-null    float64
8   RAD         506 non-null    int64
9   TAX         506 non-null    int64
10  PTRATIO     506 non-null    float64
11  B           506 non-null    float64
12  LSTAT       486 non-null    float64
13  MEDV        506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  -
0   CRIM        506 non-null    float64
1   ZN          506 non-null    float64
2   INDUS       506 non-null    float64
3   CHAS        506 non-null    float64
4   NOX         506 non-null    float64
5   RM          506 non-null    float64
6   AGE         506 non-null    float64
7   DIS         506 non-null    float64

```



```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 506 entries, 0 to 505
```

```
Data columns (total 14 columns):
```

#	Column	Non-Null Count	Dtype
0	CRIM	506 non-null	float64
1	ZN	506 non-null	float64
2	INDUS	506 non-null	float64
3	CHAS	506 non-null	float64
4	NOX	506 non-null	float64
5	RM	506 non-null	float64
6	AGE	506 non-null	float64
7	DIS	506 non-null	float64
8	RAD	506 non-null	int64
9	TAX	506 non-null	int64
10	PTRATIO	506 non-null	float64
11	B	506 non-null	float64
12	LSTAT	506 non-null	float64
13	MEDV	506 non-null	float64

```
dtypes: float64(12), int64(2)
```

```
memory usage: 55.5 KB
```

```

[CV] END .....n_estimators=400; total time= 0.3s
[CV] END .....n_estimators=400; total time= 0.3s
[CV] END .....n_estimators=400; total time= 0.4s
[CV] END .....n_estimators=500; total time= 0.4s
[CV] END .....n_estimators=500; total time= 0.4s
[CV] END .....n_estimators=500; total time= 0.4s
[CV] END .....n_estimators=500; total time= 0.4s
[CV] END .....n_estimators=500; total time= 0.4s
[CV] END .....n_estimators=600; total time= 0.5s
[CV] END .....n_estimators=600; total time= 0.5s
[CV] END .....n_estimators=600; total time= 0.5s
[CV] END .....n_estimators=600; total time= 0.5s
[CV] END .....n_estimators=600; total time= 0.5s
[CV] END .....n_estimators=700; total time= 0.6s
[CV] END .....n_estimators=700; total time= 0.6s
[CV] END .....n_estimators=700; total time= 0.6s
[CV] END .....n_estimators=700; total time= 0.6s
[CV] END .....n_estimators=700; total time= 0.6s
[CV] END .....n_estimators=800; total time= 0.7s
[CV] END .....n_estimators=800; total time= 0.7s
[CV] END .....n_estimators=800; total time= 0.7s
[CV] END .....n_estimators=800; total time= 0.7s
[CV] END .....n_estimators=800; total time= 0.7s
[CV] END .....n_estimators=900; total time= 0.8s
[CV] END .....n_estimators=900; total time= 0.8s
[CV] END .....n_estimators=900; total time= 0.8s
[CV] END .....n_estimators=900; total time= 0.8s
[CV] END .....n_estimators=900; total time= 0.8s
[CV] END .....n_estimators=1000; total time= 0.9s
[CV] END .....n_estimators=1000; total time= 0.8s
[CV] END .....n_estimators=1000; total time= 0.9s
[CV] END .....n_estimators=1000; total time= 0.8s
[CV] END .....n_estimators=1000; total time= 0.8s
Random Forest accuracy is --> 97.93621342653312

```

**Conclusion:**