**Practical 1**

**Aim: Develop Programs to understand the control structure of python.**

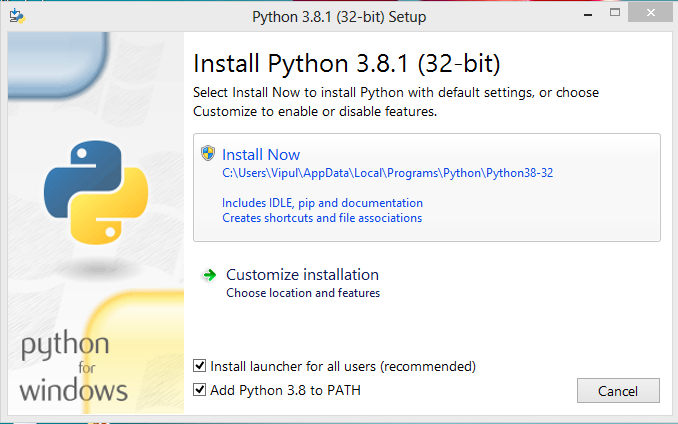
**(a) Write steps to install python and pycharm IDE for Windows Python Programming.**

**Steps of installing python:**

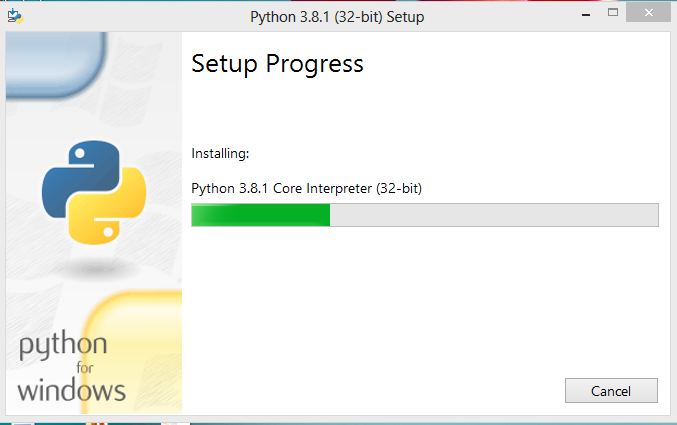
**Step 1)** To download and install Python visit the official website of Python http://www.python.org/downloads/ and choose your version. We have chosen Python version 3.6.3



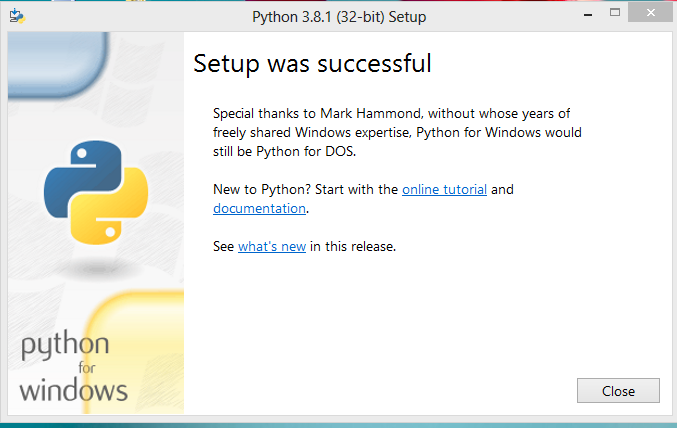
**Step 2)** Once the download is complete, run the exe for install Python. Now click on Install Now.



**Step 3)** You can see Python is installing at this point.

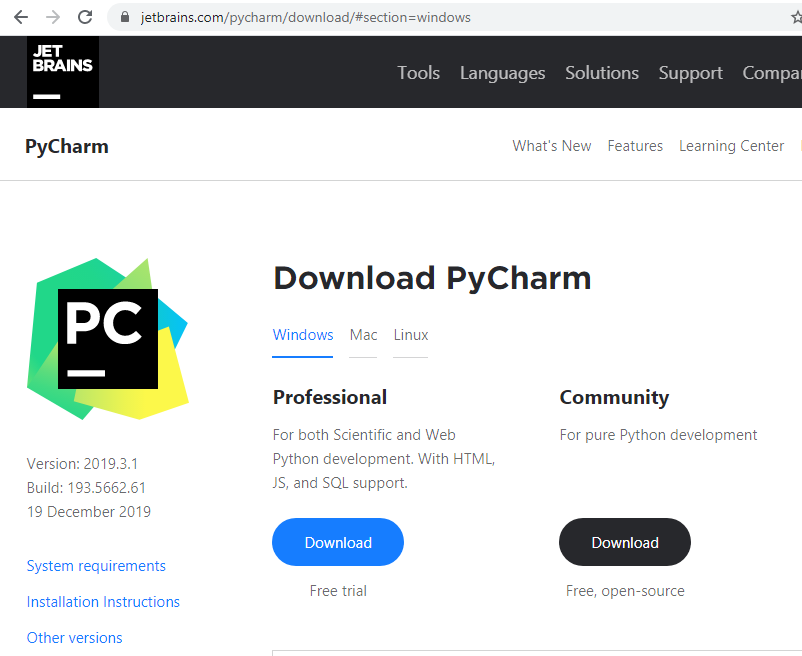


**Step 4)** When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

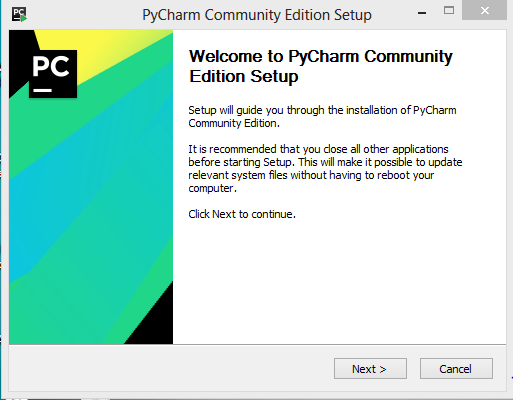


**Steps of installing pycharm:**

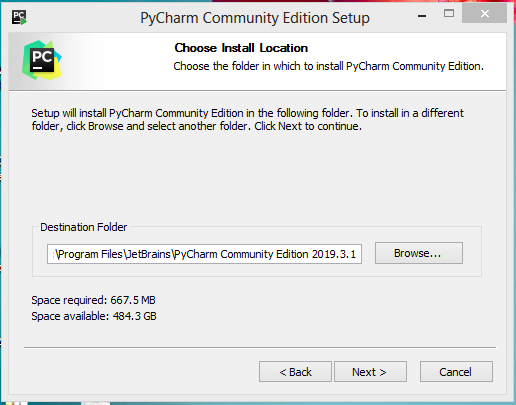
**Step 1)** To download PyCharm visit the website https://www.jetbrains.com/pycharm/download/ and Click the "DOWNLOAD" link under the Community Section.



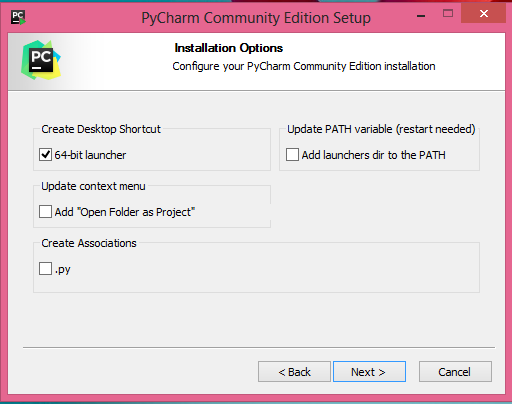
**Step 2)** Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.



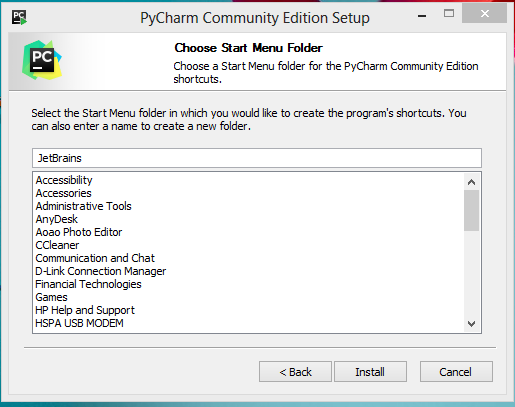
**Step 3)** On the next screen, Change the installation path if required. Click “Next”.



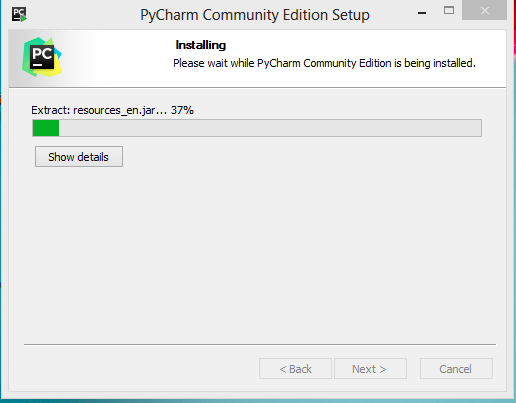
**Step 4)** On the next screen, you can create a desktop shortcut if you want and click on “Next”.



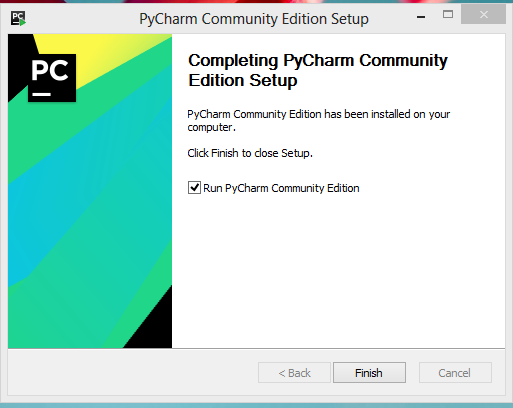
**Step 5)** Choose the start menu folder. Keep selected JetBrains and click on “Install”.



**Step 6)** Wait for the installation to finish.



**Step 7)** Once installation is finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.



**(b) Write a program to implement all arithmetic operators.**

num1=eval(input("Enter value of num1: "))  
num2=eval(input("Enter value of num2: "))  
def add(a,b):  
 return a+b  
def sub(a,b):  
 return a-b  
def mul(a,b):  
 return a\*b  
def div(a,b):  
 return a/b  
print("select operation")  
print("1.Addition")  
print("2.Subtraction")  
print("3.Multiplication")  
print("4.Division")  
choice=eval(input("Enter the choice: "))  
print(choice)  
if choice==1:  
 print("Addition is: ",add(num1,num2))  
elif choice==2:  
 print("Subtraction is: ",sub(num1,num2))  
elif choice==3:  
 print("Multiplication is: ",mul(num1,num2))  
elif choice==4:  
 print("Division is: ",div(num1,num2))  
else:  
 print("Invalid Input")

**Output:**

Enter value of num1: 20

Enter value of num2: 10

select operation

1.Addition

2.Subtraction

3.Multiplication

4.Division

Enter the choice: 1

1

Addition is: 30

Enter value of num1: 20

Enter value of num2: 10

select operation

1.Addition

2.Subtraction

3.Multiplication

4.Division

Enter the choice: 2

2

Subtraction is: 10

Enter value of num1: 20

Enter value of num2: 10

select operation

1.Addition

2.Subtraction

3.Multiplication

4.Division

Enter the choice: 3

3

Multiplication is: 200

Enter value of num1: 20

Enter value of num2: 10

select operation

1.Addition

2.Subtraction

3.Multiplication

4.Division

Enter the choice: 4

4

Division is:2.0

Enter value of num1: 20

Enter value of num2: 10

select operation

1.Addition

2.Subtraction

3.Multiplication

4.Division

Enter the choice: 5

5

Invalid Input

**(c) Write a program to find the largest out of three numbers given by users.**

num1=eval(input("Enter value of num1: "))  
num2=eval(input("Enter value of num2: "))  
num3=eval(input("Enter value of num3: "))  
  
def max(a,b,c):  
 if (a>b and a>c):  
 return a   
 elif (c>a and c>b):  
 return c  
 else:  
 return b  
print("Largest number is: ",max(num1,num2,num3))

**Output:**

Enter value of num1: 43

Enter value of num2: 23

Enter value of num3: 65

Largest number is: 65

**(d) Display all prime numbers within range given by users.**

lower=eval(input("Enter the lower Number:"))  
higher=eval(input("Enter the higher Number:"))  
l=1  
for num in range(lower,higher+1):  
 if(num>0):  
 for i in range(2,num):  
 if(num%i==0):  
 l=1  
 break  
 if(l==1):  
 l=0  
 else:   
 print("prime numbers are: ",num)

**Output:**

Enter the lower Number:1

Enter the higher Number:20

prime numbers are: 2

prime numbers are: 3

prime numbers are: 5

prime numbers are: 7

prime numbers are: 11

prime numbers are: 13

prime numbers are: 17

prime numbers are: 19

**Practical 2**

**Aim: Develop programs to learn different types of structures(list, dictionary, tuples) in python.**

**(a) Create a list(list1) of ten elements. Separate the elements of list1 into two lists say(list2 and list3) , one for storing even and other for odd from list1.**

list1=[ ]

n=int(input("enter the number of elements: "))

for i in range(1,n+1):

b=int(input("enter element: "))

list1.append(b)

list2=[ ]

list3=[ ]

for j in list1:

if(j%2==0):

list2.append(j)

else:

list3.append(j)

print("the even list: ",list2)

print("the odd list: ",list3)

**Output:**

enter the number of elements: 10

enter element: 23

enter element: 45

enter element: 68

enter element: 2

enter element: 6

enter element: 4

enter element: 31

enter element: 80

enter element: 57

enter element: 91

the even list: [68, 2, 6, 4, 80]

the odd list: [23, 45, 31, 57, 91]

**(b) Create list1 and list2. Copy the content in list3.**

list1=input("enter the content of list 1: \n")

list2=input("enter the content of list 2: \n")

list1=list1.split(' ')

list2=list2.split(' ')

list3=[ ]

list3.extend(list1)

list3.extend(list2)

print(list1)

print(list2)

print(list3)

**Output:**

enter the content of list 1:

A programming language is a formal language, which consist of a set of instructions that produce various kinds of output.

enter the content of list 2:

Programming languages are used to implement algorithms.

['A', 'programming', 'language', 'is', 'a', 'formal', 'language,', 'which', 'consist', 'of', 'a', 'set', 'of', 'instructions', 'that', 'produce', 'various', 'kinds', 'of', 'output.']

['Programming', 'languages', 'are', 'used', 'to', 'implement', 'algorithms.']

['A', 'programming', 'language', 'is', 'a', 'formal', 'language,', 'which', 'consist', 'of', 'a', 'set', 'of', 'instructions', 'that', 'produce', 'various', 'kinds', 'of', 'output.', 'Programming', 'languages', 'are', 'used', 'to', 'implement', 'algorithms.']

**(c) Create two 3x3 matrix using list and store addition in the result matrix.**

A=[ ]

n=int(input("Enter size for matrix: "))

print("Enter the elements : ")

for i in range(n):

row=[ ]

for j in range(n):

row.append(int(input()))

A.append(row)

print("matrix 1: \n")

for i in range(n):

for j in range(n):

print(A[i][j], end=" ")

print()

B=[ ]

n=int(input("Enter size for matrix : "))

print("Enter the elements : ")

for i in range(n):

row=[]

for j in range(n):

row.append(int(input()))

B.append(row)

print("matrix 2: \n")

for i in range(n):

for j in range(n):

print(B[i][j], end=" ")

print()

result = [[0,0,0], [0,0,0], [0,0,0]]

print("Resultant Matrix is : ")

for i in range(n):

for j in range(len(A[0])):

result[i][j] = A[i][j] + B[i][j]

for r in result:

print(r)

**Output:**

Enter size for matrix: 3

Enter the elements :

1

3

5

7

9

2

4

6

8

matrix 1:

1 3 5

7 9 2

4 6 8

Enter size for matrix : 3

Enter the elements :

2

4

6

8

1

3

5

7

9

matrix 2:

2 4 6

8 1 3

5 7 9

Resultant Matrix is :

[3, 7, 11]

[15, 10, 5]

[9, 13, 17]

**(d) Write a program to demonstrate class concepts along with constructor and destructor.**

class abc:

def \_\_init\_\_(self,name,year,private):

self.name=name

self.year=year

self.\_\_private=private

def \_\_del\_\_(self):

print("object with value %d is going out of scope" %self.year)

def \_\_repr\_\_(self):

return repr(self.name)

def show(self):

print(self.name)

print(self.year)

print(self.\_\_private)

n1=abc("hello",123,16)

n1.show()

print("repr",repr(n1))

n2=abc("Object 2",456,12)

n2.show()

del n1

del n2

**Output:**

hello

123

16

repr 'hello'

Object 2

456

12

object with value 123 is going out of scope

object with value 456 is going out of scope

**(e) Demonstrate use of dictionary and all its functions which can be operated on dictionary(i.e len(),itmes(),keys(),values() etc.....)**

d1={"abc":98,"xyz":97,"pqr":95}

d2={}

print("Length of the dictionary: ",len(d1))

print(d1.keys())

print(d1.values())

print(d1.items())

del d1["pqr"]

print("After deleting: ",d1)

print("Copied dictionary: ",d1.copy())

print("Pop last elements: ",d1.popitem())

print("Get value by key: ",d1.get("abc"))

**Output:**

Length of the dictionary: 3

dict\_keys(['abc', 'xyz', 'pqr'])

dict\_values([98, 97, 95])

dict\_items([('abc', 98), ('xyz', 97), ('pqr', 95)])

After deleting: {'abc': 98, 'xyz': 97}

Copied dictionary: {'abc': 98, 'xyz': 97}

Pop last elements: ('xyz', 97)

Get value by key: 98

**Practical 3**

**Aim: Develop program to learn concept of functions scooping, recursion and list mutability.**

**(a) Write a program to generate Fibonacci series using recursion.**

ctr=0

def fibo(a,b,crt):

c=a+b

print(c)

a=b

b=c

global ctr

ctr=ctr+1

if(ctr<10):

fibo(a,b,ctr)

else:

return c

print(0)

print(1)

fibo(0,1,ctr)

**Output:**

0

1

1

2

3

5

8

13

21

34

55

89

**(b) write a program to create a calculator(for each operator keep separate function)**

**Pract3\_b1:**

import math

def add(a,b):

return a+b

def sub(a,b):

return a-b

def mul(a,b):

return a\*b

def div(a,b):

return a/b

def mod(a,b):

return a%b

def rec(a):

return (1/a)

def nigate(a):

return (-a)

def sqrt(a):

return (math.sqrt(a))

**Pract3\_b2:**

import pract3\_b1

a=int(input("Enter a: "))

b=int(input("Enter b: "))

print(pract3\_b1.add(a, b))

print(pract3\_b1.sub(a, b))

print(pract3\_b1.mul(a, b))

print(pract3\_b1.div(a, b))

print(pract3\_b1.mod(a, b))

print(pract3\_b1.rec(a))

print(pract3\_b1.nigate(a))

print(pract3\_b1.sqrt(a))

**Output:**

Enter a: 2

Enter b: 5

7

-3

10

0.4

2

0.5

-2

1.4142135623730951

**Practical 4**

**Aim: Develop programs to understand working of exception handling**

**(a) Write a program to handle file opening(file not found) and divide by zero exception.**

ifile=0

a=int(input("Enter number1: "))

b=int(input("Enter number2: "))

try:

print(a/b)

except ZeroDivisionError:

print("divid by 0")

else:

while(ifile==0):

try:

i=input("Enter file name: ")

file=open(i,"r")

except FileNotFoundError:

print("File could not be found")

else:

for line in file:

print(line)

ifile=1

finally:

if(ifile==1):

print("File operation completed successfully")

else:

print("Enter correct file name")

**Output:**

Enter number1: 4

Enter number2: 2

2.0

Enter file name: file.txt

hello, how are you ??

good morning

File operation completed successfully

**(b) Write a program to handle exception generated due to immutability of tuple element.**

try:

t=('j',19,5,'r')

t[2]='k'

except ZeroDivisionError:

print("divide by zero error")

except TypeError:

print("Tuple is immutable")

else:

print(t)

**Output:**

Tuple is immutable

**Practical 5**

**Aim: Develop programs for data structure algorithms using python –searching, sorting and has tables.**

**(a) W.A.P to perform selection sort.**

n=int(input("Enter the number of elements: "))

l=[]

for i in range(n):

a=input("Enter the elements: ")

l.append(a)

print("Before sorting: ",l)

def selsort(l):

s=0

while s!=len(l):

for i in range(s,len(l)):

if l[s]>l[i]:

l[s],l[i]=l[i],l[s]

s+=1

return l

print("After sorting: ",selsort(l))

**Output:**

Enter the number of elements: 5

Enter the elements: 5

Enter the elements: 4

Enter the elements: 6

Enter the elements: 3

Enter the elements: 2

Before sorting: ['5', '4', '6', '3', '2']

After sorting: ['2', '3', '4', '5', '6']

**(b) W.A.P to merge sort.**

n=int(input("Enter the number of elements: "))

l=[]

for i in range(1,n+1):

a=input("Enter the elements: ")

l.append(a)

print("Before sorting: ",l)

def mergesort(l):

if len(l)>1:

mid=len(l)//2

left=l[:mid]

right=l[mid:]

mergesort(left)

mergesort(right)

i=j=k=0

while(i<len(left) and j<len(right)):

if left[i]<right[j]:

l[k]=left[i]

i=i+1

else:

l[k]=right[j]

j=j+1

k=k+1

while i<len(left):

l[k]=left[i]

i=i+1

k=k+1

while j<len(right):

l[k]=right[j]

j=j+1

k=k+1

return l

print("After sorting: ",mergesort(l))

**Output:**

Enter the number of elements: 5

Enter the elements: 6

Enter the elements: 8

Enter the elements: 4

Enter the elements: 2

Enter the elements: 5

Before sorting: ['6', '8', '4', '2', '5']

After sorting: ['2', '4', '5', '6', '8']

**(c) W.A.P to do binary search on sorted elements.**

import math

n = int(input("Enter the number of elements: "))

arr = []

for i in range(1, n + 1):

a = int(input("Enter the elements: "))

arr.append(a)

print("sorted elements: ", sorted(arr))

x = int(input("Enter the value of x: "))

def binarysearch(arr, l, r, x):

if r>=1:

mid =math.floor(l+(r-l)/2)

if(arr[mid] == x):

return mid

elif(arr[mid]>x):

return binarysearch(arr,l,mid-1,x)

else:

return binarysearch(arr,mid+1,r,x)

else:

return -1

result = binarysearch(arr, 0, len(arr)-1, x)

if result != -1:

print("Element is present at index %d" % result)

else:

print("Element is not present in list.")

**Output:**

Enter the number of elements: 5

Enter the elements: 4

Enter the elements: 3

Enter the elements: 5

Enter the elements: 1

Enter the elements: 2

sorted elements: [1, 2, 3, 4, 5]

Enter the value of x: 4

Element is present at index 3

**Practical 6**

**Aim: Develop programs to understand concepts of threading:**

**(a) Demonstrate the custom thread and use of join function.**

import threading

def add(x,y,z):

print("addition:{}".format(x+y+z))

def sub(x,y,z):

print("\n subtraction:{}".format(x-y-z))

def mul(x,y,z):

print("\n multiplication:{}".format(x\*y\*z))

def div(x,y,z):

print("\n division:{}".format(x/y/z))

if \_\_name\_\_ == "\_\_main\_\_":

t1=threading.Thread(target=add, args=(10,20,30,))

t2=threading.Thread(target=sub, args=(10,20,30,))

t3=threading.Thread(target=mul, args=(10,20,30,))

t4=threading.Thread(target=div, args=(10,20,30,))

t1.start()

t2.start()

t3.start()

t4.start()

t1.join()

t2.join()

t3.join()

t4.join()

**Output:**

addition:60

subtraction:-40

division:0.016666666666666666

multiplication:6000

**(b) Demonstrate the use of lock for threading.**

import threading

x=0

def increment():

global x

x+=1

def thread\_task(Lock):

for \_ in range(100000):

Lock.acquire()

increment()

Lock.release()

def main\_task():

global x

x=0

lock1=threading.Lock()

t1=threading.Thread(target=thread\_task, args=(lock1,))

t2=threading.Thread(target=thread\_task, args=(lock1,))

t1.start()

t2.start()

t1.join()

t2.join()

if \_\_name\_\_ == "\_\_main\_\_" :

for i in range(10):

main\_task()

print("iteration{0}:x={1}".format(i,x))

**Output:**

iteration0:x=200000

iteration1:x=200000

iteration2:x=200000

iteration3:x=200000

iteration4:x=200000

iteration5:x=200000

iteration6:x=200000

iteration7:x=200000

iteration8:x=200000

iteration9:x=200000

**Practical 7**

# Aim: Develop program for socket programming in Python.

**(a) Write a program to perform TCP server and client. Server:**

import socket

def Main():

host='127.0.0.1'

port=5000

s=socket.socket()

s.bind((host,port))

s.listen(1)

c,addr=s.accept()

print "Connection from:" +str(addr)

print c

while True:

data=c.recv(1024)

if not data:

break

print"from connected user:"+str(data)

data=str(data).upper()

print "sending :" + str(data)

c.send(data)

c.close()

if \_\_name\_\_=='\_\_main\_\_':

Main()

# Client:

import socket

def Main():

host='127.0.0.1'

port=5000

s=socket.socket()

s.connect((host,port))

mess=raw\_input("Enter message:")

while mess!='q':

s.send(mess)

data=s.recv(1024)

print "Received from server:"+str(data)

mess=raw\_input("Enter data:")

s.close()

if \_\_name\_\_=='\_\_main\_\_':

Main()

## Output:

**Server:**

Connection from: ('127.0.0.1', 50057)

from connected user : hello

sending : HELLO

## Client:

Enter data: hello

Received from server: HELLO

Enter data: q

Connection Terminated

# (b) Write a program to perform UDP server and client.

**Server:**

import socket

UDP\_IP ="localhost"

UDP\_PORT = 8080

MESSAGE = input('Send message : ')

print("message:",MESSAGE)

print("message sent")

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

sock.sendto(bytes(MESSAGE, "utf-8"), (UDP\_IP, UDP\_PORT))

# Client:

import socket

UDP\_IP = "localhost"

UDP\_PORT = 8080

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

sock.bind((UDP\_IP, UDP\_PORT))

while True:

*# buffer size is 1024 bytes*

data, addr = sock.recvfrom(1024)

print("Received message:", data.decode())

## Output:

**Server:**

Send message : hello

message: hello

message sent

## Client:

Received message:hello

**Practical 8**

**Aim: Demonstrate various functions of turtle**

import turtle

a=turtle.Pen()

a.shape("turtle")

a.speed(10)

a.color("red")

a.width(5)

a.forward(100)

a.reset()

a.circle(100)

a.circle(-100)

a.reset()

a.color("blue")

a.forward(100)

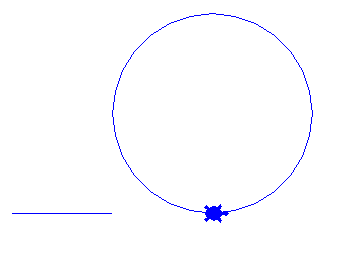
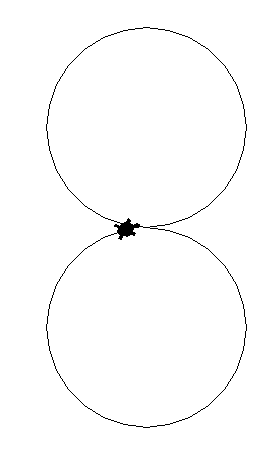
a.up()

a.forward(100)

a.down()

a.circle(100)

**Output:**

**Practical 9**

**Aim: Develop program to demonstrate Tkinter for GUI. Demonstrate the use of radio button and button using Tkinter.**

from tkinter import \*

master = Tk()

master.title("TKINTER EXAMPLE")

Label(master).grid(row=0)

Label(master, text='First Name').grid(row=1)

Label(master, text='Last Name').grid(row=2)

e1 = Entry(master)

e2 = Entry(master)

e1.grid(row=1, column=1)

e2.grid(row=2, column=1)

v = IntVar()

Radiobutton(master, text='Male', variable=v, value=1).grid(row=4,column=0)

Radiobutton(master, text='Female', variable=v, value=2).grid(row=4,column=1)

Button(master, bg='green',fg='white',text='Submit', width=25, ).grid(row=5)

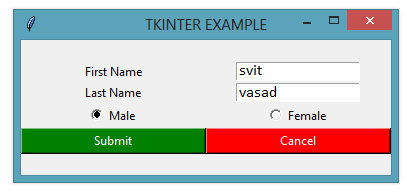
Button(master, bg='red',fg='white',text='Cancel', width=25,

command=master.destroy).grid(row=5,column=1)

Label(master, text=e1.get()).grid(row=6)

mainloop()

**Output:**



**Practical 10**

**Aim: Learn to plot different types of graphs using pyplot.**

**(a) Simple Plot**

import matplotlib.pyplot as plt

x1 = [1,2,3]

y1 = [2,4,1]

plt.plot(x1, y1, label = "line 1")

x2 = [1,2,3]

y2 = [4,1,3]

plt.plot(x2, y2, label = "line 2")

plt.xlabel('x - axis')

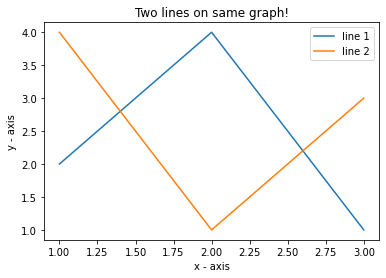
plt.ylabel('y - axis')

plt.title('Two lines on same graph!')

plt.legend()

plt.show()

**Output:**



**(b) Scatter Plot**

import matplotlib.pyplot as plt

x = [1,2,3,4,5,6,7,8,9,10]

y = [2,4,5,7,6,8,9,11,12,12]

plt.scatter(x, y, label= "stars", color= "green",marker= "\*", s=30)

plt.xlabel('x - axis')

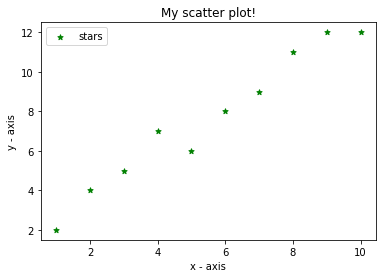
plt.ylabel('y - axis')

plt.title('My scatter plot!')

plt.legend()

plt.show()

**Output:**

****

**(c) Bar Graph Plot**

import matplotlib.pyplot as plt

left = [1, 2, 3, 4, 5]

height = [10, 24, 36, 40, 5]

tick\_label = ['one', 'two', 'three', 'four', 'five']

plt.bar(left, height, tick\_label = tick\_label,width = 0.8, color = ['red', 'green'])

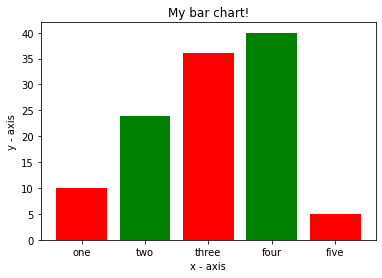
plt.xlabel('x - axis')

plt.ylabel('y - axis')

plt.title('My bar chart!')

plt.show()

**Output:**



**(d) Histogram Plot**

import matplotlib.pyplot as plt

ages = [2,5,70,40,30,45,50,45,43,40,44,60,7,13,57,18,90,77,32,21,20,40]

range = (0, 100)

bins = 10

plt.hist(ages, bins, range, color = 'green',histtype = 'bar', rwidth = 0.8)

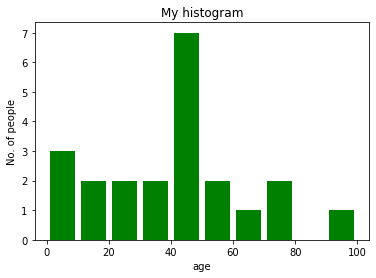
plt.xlabel('age')

plt.ylabel('No. of people')

plt.title('My histogram')

plt.show()

**Output:**



**(e) Pie Plot**

import matplotlib.pyplot as plt

activities = ['eat', 'sleep', 'work', 'play']

slices = [3, 7, 8, 6]

colors = ['r', 'y', 'g', 'b']

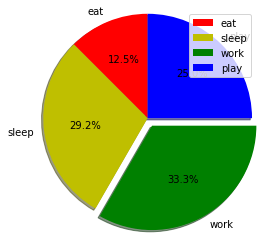
plt.pie(slices, labels = activities, colors=colors,startangle=90, shadow = True, explode = (0, 0, 0.1, 0),

radius = 1.2, autopct = '%1.1f%%')

plt.legend()

plt.show()

**Output:**



**Practical 11**

**Aim: Implement classical ciphers using python.**

def encrypt(text, s):

result = ""

for i in range(len(text)):

char = text[i]

if (char.isupper()):

result += chr((ord(char) + s - 65) % 26 + 65)

else:

result += chr((ord(char) + s - 97) % 26 + 97)

return result

text = input("Enter text: ")

s=int(input("enter key: "))

print("Text : " ,text)

print("Shift : " ,str(s))

print("Cipher: " , encrypt(text, s) )

**Output:**

Enter text: apqyz

enter key: 2

Text : apqyz

Shift : 2

Cipher: crtab

**Beyond Syllabus Practical**

**Aim: Implement Linear regression technique on boston\_house dataset in python.**

# -\*- coding: utf-8 -\*-

"""beyond\_linear\_reg.ipynb

Automatically generated by Colaboratory.

"""

# import all the important libraries.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# %matplotlib inline

import sklearn

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn import metrics

from sklearn.metrics import r2\_score

# load the boston dataset from the sklearn library.

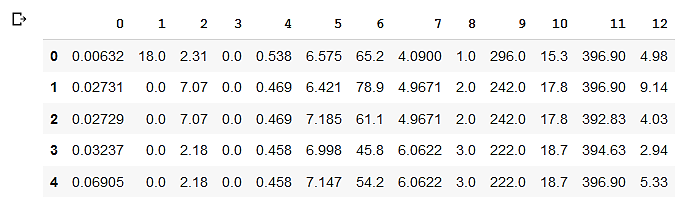
from sklearn.datasets import load\_boston

boston = load\_boston()

# load the data into a pandas dataframe and then will print the first few rows of the data

bos = pd.DataFrame(boston.data)

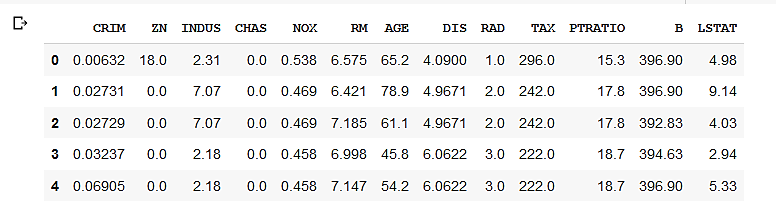
bos.head()



# rename the columns as the description of the dataset given above.

bos.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT']

bos.head()



# The variable MEDV indicates the prices of the houses and is the target variable.

# The rest of the variables are the predictors based on which we will predict the value of the house.

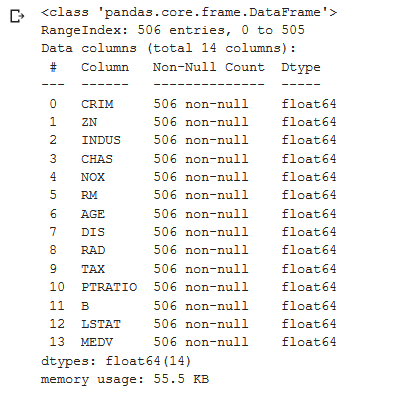
# In the above result, we can see that the target variable ‘MEDV’ is missing from the data.

# We will create a new column of target values and add them to the dataframe.

bos['MEDV'] = boston.target

# fetching more information about the dataset

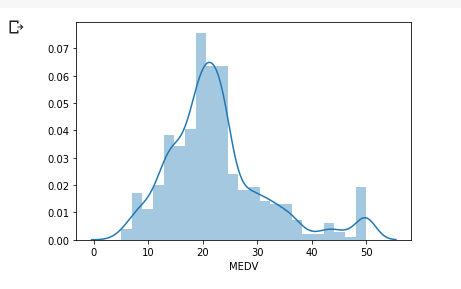
bos.info()



# understand the relationship of the target variable with other variables using Exploratory Data Analysis(EDA)

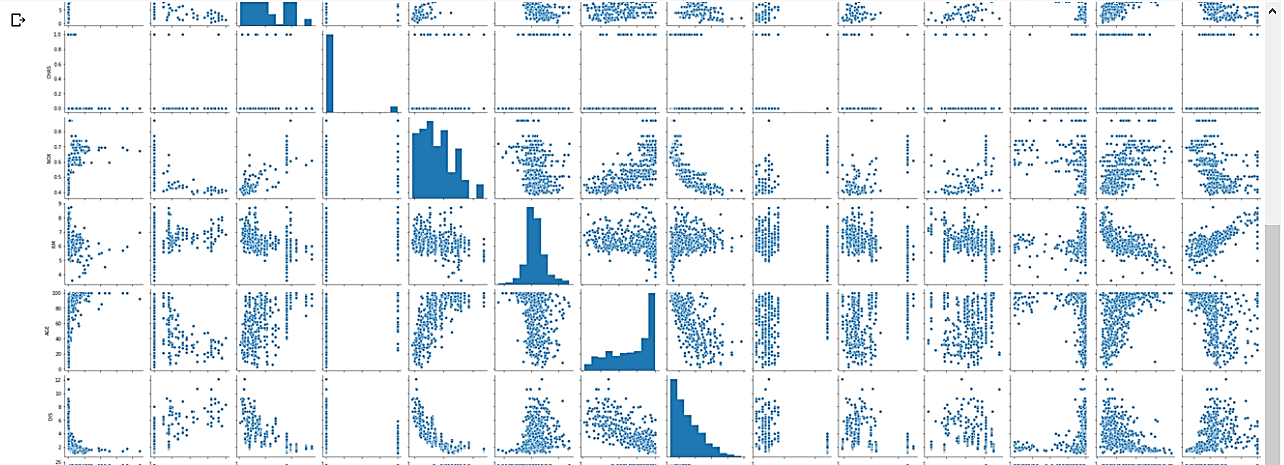
sns.distplot(bos['MEDV'])

plt.show()



# visualize the pairplot which shows the relationships between all the features present in the dataset.

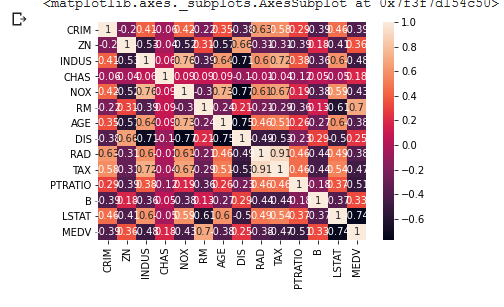
sns.pairplot(bos)



# use the heatmap function from the seaborn library to plot the correlation matrix.

corr\_mat = bos.corr().round(2)

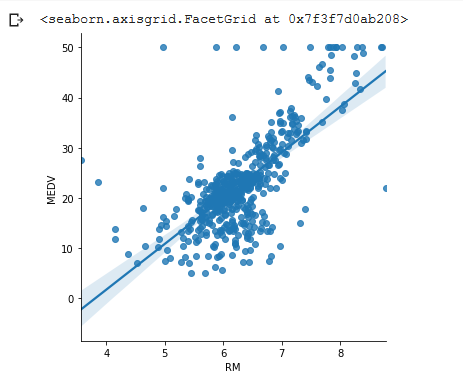
sns.heatmap(data=corr\_mat, annot=True)



# feature RM has a positive correlation with MEDV from above two plots.

# plot an lmplot between RM and MEDV to see the relationship between the two more clearly.

sns.lmplot(x = 'RM', y = 'MEDV', data = bos)



# split the dataset into training and test data

# train our model with 80% of the samples and test with the remaining 20%

X = bos[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX','PTRATIO', 'B', 'LSTAT']]

y = bos['MEDV']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 10)

# train our model using the LinearRegression function from the sklearn library

lm = LinearRegression()

lm.fit(X\_train, y\_train)

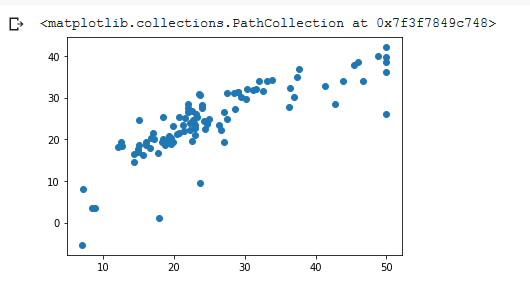


# make prediction on the test data using the LinearRegression function

# plot a scatterplot between the test data and the predicted value

prediction = lm.predict(X\_test)

plt.scatter(y\_test, prediction)

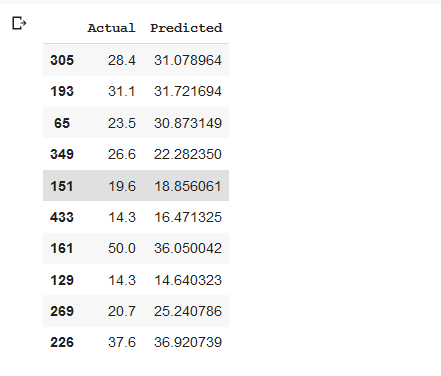


# Plotting the data frame for the actual and predicted value

df1 = pd.DataFrame({'Actual': y\_test, 'Predicted':prediction})

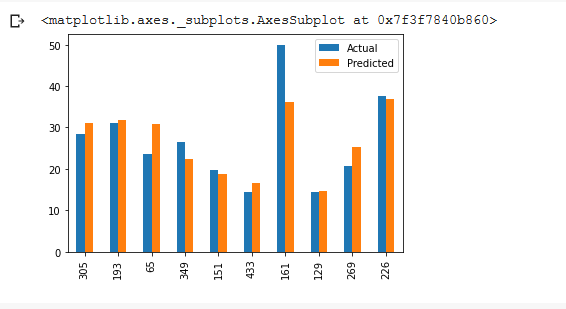
df2 = df1.head(10)

df2



# plotting graph

df2.plot(kind = 'bar')



# evaluate the model

print('MAE', metrics.mean\_absolute\_error(y\_test, prediction))

print('MSE', metrics.mean\_squared\_error(y\_test, prediction))

print('RMSE', np.sqrt(metrics.mean\_squared\_error(y\_test, prediction)))

print('R squared error', r2\_score(y\_test, prediction))

