**DATA SCIENCE WITH PYTHON**

Student Name:M.Omkar

Id Number:N190706

Class:CSE-03

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINERING**

**RGUKT-NUZ-AP E2-SEMISTER-II**

**AY-2022-23**

**Table of Contents :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Assignment No** | **Assignment Name** | **Page Number** | **Date** |
| 1 | Python Basics | 3-5 |  |
| 2 | Python Data Structures | 6-11 |  |
| 3 | Python programming fundamentals | 12-16 |  |
| 4 | Working with data in python | 17-21 |  |
| 5 | Working Numpy Arrays | 22-26 |  |
| 6 | Importing datasets and Cleaning and preparing the Data | 27-32 |  |
| 7 | Model Development | 33-38 |  |
| 8 | Model Evalution | 39-43 |  |
| 9 | Introduction to visualization tools | 44-45 |  |
| 10 | Basic visualization tools | 46-48 |  |
| 11 | Specialized visualization tolls | 49-52 |  |
| 12 | Advanced Visualization tools | 53-57 |  |

**ASSIGNMENT-01**

**Python Basics:** Your first program, Types Expressions and Variables String Operations

**Code:**

name='omkar'

print(name)

name='RaviTeja'

print(name)

a,b,c=5,3.2,'Hello'

print(a)

print(b)

print(c)

a=b=c=100

print(a)

print(b)

print(c)

name=fname='Omkar'

print(name)

print(fname)

 num=5

 Num=55

print(num)

print(Num)

value=None

print(value)

fruits=['apple','mango','orange']

print(fruits)

numbers=[1,2,3]

print(numbers)

alaphabets={'a':'apple','b':'ball','c':'cat'}

print(alaphabets)

vowels={'a','e','i','o','u'}

print(vowels

num1=59

print(num1,'is of type',type(num1))

num2=2.0

print(num2,'is of type',type(num2))

num3=1+2j

print(num3,'is of type',type(num3))

languages=["c","oops","python","DS"]

print(languages[0])

print(languages[3])

product=('Microsoft','Xbox',999)

print(product[0])

print(product[1])

name='omkar'

print(name)

message='study well '

print(message)

student\_id={1,2,3,4,5}

print(student\_id)

print(type(student\_id))

capital\_city={'Nepal':'kathmandu','italy':'rome','england':'london'}

print(capital\_city)

greet='hello'

print(greet[1])

print(greet[4])

print(greet[1:4])

message='Welcome to CSE'

message='welcome to cse-03'

print(message)

message="""

Never gonna give you up

Never gonna let you down

"""

print(message)

str1 = "Hello, world!"

str2 = "I love Python."

str3 = "Hello, world!"

print(str1 == str2)

print(str1 == str3)

greet = "Hello, "

name = "Jack"

result = greet + name

print(result)

greet = 'Hello'

for letter in greet:

print(letter)

greet = 'Hello'

print(len(greet))

print('a' in 'program')

print('at' not in 'battle')

example = "He said, \"What's there?\""

example = 'He said, "What\'s there?"'

print(example)

name = 'Cathy'

country = 'UK'

print(f'{name} is from {country}')

**Output:**

omkar

RaviTeja

5

3.2

Hello

100

100

100

Omkar

Omkar

5

55

None

['apple', 'mango', 'orange']

[1, 2, 3]

{'a': 'apple', 'b': 'ball', 'c': 'cat'}

{'i', 'u', 'a', 'e', 'o'}

59 is of type <class 'int'>

2.0 is of type <class 'float'>

(1+2j) is of type <class 'complex'>

c

DS

Microsoft

Xbox

omkar

study well

{1, 2, 3, 4, 5}

<class 'set'>

{'Nepal': 'kathmandu', 'italy': 'rome', 'england': 'london'}

e

o

ell

welcome to cse-03

Never gonna give you up

Never gonna let you down

False

True

Hello, Jack

H

e

l

l

o

5

True

False

He said, "What's there?"

Cathy is from UK

**ASSIGNMENT-02**

**Python Data Structures:** Lists and Tuples Sets,and Dictionaries

**Code:**

numbers = [1, 2, 5]

print(numbers)

languages = ["Python", "Swift", "C++"]

print(languages[0])

print(languages[2])

languages = ["Python", "Swift", "C++"]

print(languages[-1])

print(languages[-3])

my\_list = ['p','r','o','g','r','a','m','i','z']

print(my\_list[2:5])

print(my\_list[5:])

print(my\_list[:])

numbers = [21, 34, 54, 12]

print("Before Append:", numbers)

numbers.append(32)

print("After Append:", numbers)

prime\_numbers = [2, 3, 5]

print("List1:", prime\_numbers)

even\_numbers = [4, 6, 8]

print("List2:", even\_numbers)

prime\_numbers.extend(even\_numbers)

print("List after append:", prime\_numbers)

languages = ['Python', 'Swift', 'C++']

# changing the third item to 'C'

languages[2] = 'C'

print(languages)

languages = ['Python', 'Swift', 'C++', 'C', 'Java', 'Rust', 'R']

# deleting the second item

del languages[1]

print(languages) # ['Python', 'C++', 'C', 'Java', 'Rust', 'R']

# deleting the last item

del languages[-1]

print(languages) # ['Python', 'C++', 'C', 'Java', 'Rust']

# delete first two items

del languages[0 : 2] # ['C', 'Java', 'Rust']

print(languages)

languages = ['Python', 'Swift', 'C++', 'C', 'Java', 'Rust', 'R']

# remove 'Python' from the list

languages.remove('Python')

print(languages)

languages = ['Python', 'Swift', 'C++']

# iterating through the list

for language in languages:

print(language)

languages = ['Python', 'Swift', 'C++']

print('C' in languages)

print('Python' in languages)

languages = ['Python', 'Swift', 'C++']

print("List: ", languages)

print("Total Elements: ", len(languages))

numbers = [number\*number for number in range(1, 6)]

print(numbers)

my\_tuple = ()

print(my\_tuple)

# Tuple having integers

my\_tuple = (1, 2, 3)

print(my\_tuple)

# tuple with mixed datatypes

my\_tuple = (1, "Hello", 3.4)

print(my\_tuple)

# nested tuple

my\_tuple = ("mouse", [8, 4, 6], (1, 2, 3))

print(my\_tuple)

var1 = ("hello")

print(type(var1)) # <class 'str'>

# Creating a tuple having one element

var2 = ("hello",)

print(type(var2)) # <class 'tuple'>

# Parentheses is optional

var3 = "hello",

print(type(var3))

letters = ("p", "r", "o", "g", "r", "a", "m", "i", "z")

print(letters[0])

print(letters[5])

letters = ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

print(letters[-1])

print(letters[-3])

my\_tuple = ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

# elements 2nd to 4th index

print(my\_tuple[1:4]) # prints ('r', 'o', 'g')

# elements beginning to 2nd

print(my\_tuple[:-7]) # prints ('p', 'r')

# elements 8th to end

print(my\_tuple[7:]) # prints ('i', 'z')

# elements beginning to end

print(my\_tuple[:]) # Prints ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

my\_tuple = ('a', 'p', 'p', 'l', 'e',)

print(my\_tuple.count('p'))

print(my\_tuple.index('l'))

languages = ('Python', 'Swift', 'C++')

# iterating through the tuple

for language in languages:

print(language)

languages = ('Python', 'Swift', 'C++')

print('C' in languages)

print('Python' in languages)

empty\_set = set()

# create an empty dictionary

empty\_dictionary = { }

# check data type of empty\_set

print('Data type of empty\_set:', type(empty\_set))

# check data type of dictionary\_set

print('Data type of empty\_dictionary', type(empty\_dictionary))

numbers = {2, 4, 6, 6, 2, 8}

print(numbers) # {8, 2, 4, 6

numbers = {21, 34, 54, 12}

print('Initial Set:',numbers)

# using add() method

numbers.add(32)

print('Updated Set:', numbers)

companies = {'Lacoste', 'Ralph Lauren'}

tech\_companies = ['apple', 'google', 'apple']

companies.update(tech\_companies)

print(companies)

languages = {'Swift', 'Java', 'Python'}

print('Initial Set:',languages)

# remove 'Java' from a set

removedValue = languages.discard('Java')

print('Set after remove():', languages)

fruits = {"Apple", "Peach", "Mango"}

# for loop to access each fruits

for fruit in fruits:

print(fruit)

even\_numbers = {2,4,6,8}

print('Set:',even\_numbers)

# find number of elements

print('Total Elements:', len(even\_numbers))

A = {1, 3, 5}

# second set

B = {0, 2, 4}

# perform union operation using |

print('Union using |:', A | B)

# perform union operation using union()

print('Union using union():', A.union(B))

A = {1, 3, 5}

# second set

B = {1, 2, 3}

# perform intersection operation using &

print('Intersection using &:', A & B)

# perform intersection operation using intersection()

print('Intersection using intersection():', A.intersection(B))

A = {2, 3, 5}

# second set

B = {1, 2, 6}

# perform difference operation using &

print('Difference using &:', A - B)

# perform difference operation using difference()

print('Difference using difference():', A.difference(B))

A = {2, 3, 5}

# second set

B = {1, 2, 6}

# perform difference operation using &

print('using ^:', A ^ B)

# using symmetric\_difference()

print('using symmetric\_difference():', A.symmetric\_difference(B))

A = {1, 3, 5}

# second set

B = {3, 5, 1}

# perform difference operation using &

if A == B:

print('Set A and Set B are equal')

else:

print('Set A and Set B are not equal')

capital\_city = {"Nepal": "Kathmandu", "England": "London"}

print("Initial Dictionary: ",capital\_city)

capital\_city["Japan"] = "Tokyo"

print("Updated Dictionary: ",capital\_city)

student\_id = {111: "Eric", 112: "Kyle", 113: "Butters"}

print("Initial Dictionary: ", student\_id)

student\_id[112] = "Stan"

print("Updated Dictionary: ", student\_id)

student\_id = {111: "Eric", 112: "Kyle", 113: "Butters"}

print(student\_id[111]) # prints Eric

print(student\_id[113])

student\_id = {111: "Eric", 112: "Kyle", 113: "Butters"}

print("Initial Dictionary: ", student\_id)

del student\_id[111]

print("Updated Dictionary ", student\_id)

squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

# Output: True

print(1 in squares) # prints True

print(2 not in squares) # prints True

# membership tests for key only not value

print(49 in squares) # prints false

squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

for i in squares:

print(squares[i])

**Output:**

[1, 2, 5]

Python

C++

C++

Python

['o', 'g', 'r']

['a', 'm', 'i', 'z']

['p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z']

Before Append: [21, 34, 54, 12]

After Append: [21, 34, 54, 12, 32]

<class 'list'>

List1: [2, 3, 5]

List2: [4, 6, 8]

List after append: [2, 3, 5, 4, 6, 8]

['Python', 'Swift', 'C']

['Python', 'C++', 'C', 'Java', 'Rust', 'R']

['Python', 'C++', 'C', 'Java', 'Rust']

['C', 'Java', 'Rust']

['Swift', 'C++', 'C', 'Java', 'Rust', 'R']

Python

Swift

C++

False

True

List: ['Python', 'Swift', 'C++']

Total Elements: 3

[1, 4, 9, 16, 25]

()

(1, 2, 3)

(1, 'Hello', 3.4)

('mouse', [8, 4, 6], (1, 2, 3))

<class 'str'>

<class 'tuple'>

<class 'tuple'>

p

a

z

m

('r', 'o', 'g')

('p', 'r')

('i', 'z')

('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

2

3

Python

Swift

C++

False

True

Data type of empty\_set: <class 'set'>

Data type of empty\_dictionary <class 'dict'>

{8, 2, 4, 6}

Initial Set: {34, 12, 21, 54}

Updated Set: {32, 34, 12, 21, 54}

{'Ralph Lauren', 'Lacoste', 'google', 'apple'}

Initial Set: {'Java', 'Python', 'Swift'}

Set after remove(): {'Python', 'Swift'}

Apple

Mango

Peach

Set: {8, 2, 4, 6}

Total Elements: 4

Union using |: {0, 1, 2, 3, 4, 5}

Union using union(): {0, 1, 2, 3, 4, 5}

Intersection using &: {1, 3}

Intersection using intersection(): {1, 3}

Difference using &: {3, 5}

Difference using difference(): {3, 5}

using ^: {1, 3, 5, 6}

using symmetric\_difference(): {1, 3, 5, 6}

Set A and Set B are equal

Initial Dictionary: {'Nepal': 'Kathmandu', 'England': 'London'}

Updated Dictionary: {'Nepal': 'Kathmandu', 'England': 'London', 'Japan': 'Tokyo'}

Initial Dictionary: {111: 'Eric', 112: 'Kyle', 113: 'Butters'}

Updated Dictionary: {111: 'Eric', 112: 'Stan', 113: 'Butters'}

Eric

Butters

Initial Dictionary: {111: 'Eric', 112: 'Kyle', 113: 'Butters'}

Updated Dictionary {112: 'Kyle', 113: 'Butters'}

True

True

False

1

9

25

49

81

**ASSIGNMENT-03**

**Python Programming Fundamentals:** Conditions and Branching Loops, Functions, Objects and Classes

**Code:**

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

if num>0:

print("positive")

else:

print("Negative")

i=10

if i>15:

print("i is not less than 15")

print("iam not in if")

i=20

if i<20:

print("i smaller than 15")

print("iam in if block")

else:

print("i is greatet than 15")

print("iam in else block")

def digitSum(n):

dsum=0

for ele in str(n):

dsum+=int(ele)

return dsum

list=[367,111,562,945,6726,873]

newlist=[digitSum(i) for i in list if i&1]

print(newlist)

i=10

if i==10:

if i<15:

print("i is smaller than 15")

if i<12:

print("i is smaller than 12 too")

else:

print("i is greater than 15")

i=10

if i<15:print("i is less than 15")

i=10

print(True) if i<15 else print(False)

lang=["swift","python","go","javascript"]

for i in lang:

print(i)

values=range(10)

for i in values:

print(i)

digits =[3,4,8]

for i in digits:

print(i)

else:

print("no items left")

i=1

n=6

while i<n:

print(i)

i+=1

total=0

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

while num!=0:

total+=num

num=int(input("Ente a number : "))

print(total)

age=32

while age>18:

print("you can vote")

counter=0

while counter<3:

print("inside a loop")

counter+=1

else:

print("inside else")

counter=0

while counter<3:

if counter==1:

break

print("inside a loop")

counter+=1

else:

print("inside a else")

def greet():

print("hello umashankar")

greet()

def add\_num(a,b):

return a+b

add\_num(3,5)

def find\_square(num):

r=num\*num

return r

find\_square(5)

import math

print("square of 49 is ",math.sqrt(49))

print("6 power 9 is ",math.pow(6,9))

def get\_square(n):

return n\*n

for i in [1,2,3,4,5]:

r=get\_square(i)

print("square of %d is %d"%(i,r))

def add\_numbers(a=7,b=9):

s=a+b

print(s)

add\_numbers(2,3)

#here the output will be 5

add\_numbers(4)

def display\_info(first\_name,last\_name):

print("first name is:",first\_name)

print("last name is:",last\_name)

display\_info("Omkar","Naidu")

display\_info(last\_name="Mulakala",first\_name="OmkarNaidu")

def find\_sum(\*numbers):

res=0

for i in numbers:

res+=i

print("sum is :",res)

find\_sum(1,2,3,8,6,9,9)

s=lambda n:print("even")if n%2==0 else print("odd")

s(34)

class car:

name=""

price=0

obj1=car()

obj1.name="audi"

obj1.price=10000000

print(obj1.name)

print(obj1.price)

print("type of object is :",type(obj1))

obj2=car()

obj2.name="RR"

obj2.price=100

print(obj2.name)

print(obj2.price)

class car:

def name(self,name):

print(self.name)

def price(self,price):

print(self.price)

class lorry(car):

def capacity(self,cap):

print(self.cap)

obj1=car()

obj1.name("audi")

obj1.price(120000)

class polygon:

def render(self):

print("Rendering polygon")

class square:

def render(self):

print("Rendering square")

class circle:

def render(self):

print("Rendering circle")

c=circle()

c.render()

s=square()

s.render()

p=polygon()

p.render()

obj=open("example.txt",'w+')

print(obj)

**Output:**

Enter a number: 8

positive

iam not in if

i is greatet than 15

iam in else block

[16, 3, 18, 18]

i is smaller than 15

i is smaller than 12 too

i is less than 15

True

swift

python

go

javascript

0

1

2

3

4

5

6

7

8

9

3

4

8

no items left

1

2

3

4

5

Enter a number: 4

Ente a number : 5

Ente a number : 6

Ente a number : 0

15

you can vote

you can vote

you can vote

you can vote

.

.

.

you can vote

inside a loop

inside a loop

inside a loop

inside else

inside a loop

hello umashankar

8

25

square of 49 is 7.0

6 power 9 is 10077696.0

square of 1 is 1

square of 2 is 4

square of 3 is 9

square of 4 is 16

square of 5 is 25

5

13

first name is: Omkar

last name is: Naidu

first name is: OmkarNaidu

last name is: Mulakala

sum is : 38

even

audi

10000000

type of object is : <class '\_\_main\_\_.car'>

RR

100

<bound method car.name of <\_\_main\_\_.car object at 0x7f8538e46230>>

<bound method car.price of <\_\_main\_\_.car object at 0x7f8538e46230>>

Rendering circle

Rendering square

Rendering polygon

<\_io.TextIOWrapper name='example.txt' mode='w+' encoding='UTF-8'>

**ASSIGNMENT-04**

**Working with Data in Python:** Reading files with open, Writing files with open, Loading data with Pandas, Working with and Saving data with Pandas

**Code:**

def read\_file(file\_path):

with open(file\_path, 'r') as file:

file\_contents = file.read()

return file\_contents

# Example usage

file\_path = 'example.txt' # Replace with your file path

contents = read\_file(file\_path)

print(contents)

def write\_file(file\_path, content):

with open(file\_path, 'w') as file:

file.write(content)

# Example usage

file\_path = 'example.txt' # Replace with your file path

content = 'This is the content to be written.' # Replace with the content you want to write

write\_file(file\_path, content)

import pandas as pd

df=pd.read\_csv("1.csv")

print(df)

df = pd.read\_excel('1.xlsx')

print(df)

import pandas as pd

import numpy as np

#attributes of series

s=pd.Series()

print(s)

L=[10,20,30]

s=pd.Series(L)

print(s)

print("-------------")

s={'a':10,'b':20,'c':30}

print(s)

print("-------------")

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

s=pd.Series(a,index={1,2,3})

print(s)

print("-------------")

print(s.index)

print(s.values)

print(s.name)

print(s.shape)

print(s.ndim)

print(s.size)

print(s.nbytes)

print(s.memory\_usage)

print(s.memory\_usage(index="False"))

print(s.empty)

print("-------------")

#mathematical operations

s1=pd.Series([1,2,3,4,5])

s2=pd.Series([1,2,3,4,5])

print(s1.add(s2))

print(s1.subtract(s2))

print(s1.multiply(s2))

print(s1.divide(s2))

print(s1.mod(s2))

print(s1.pow(s2))

print(s1.le(s2))

print(s1.gt(s2))

print(s1.eq(s2))

print(s1.head())

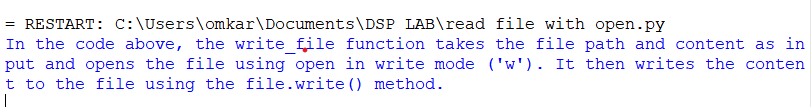
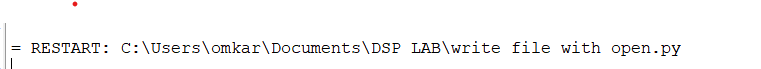
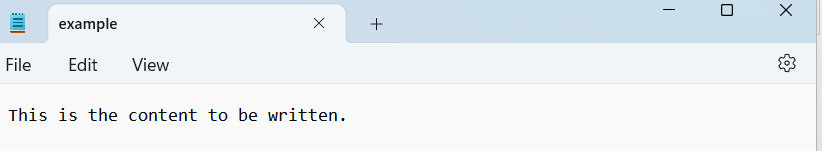
print(s1.tail())

print(s1.describe())

print(s1[0:5:2])

print("-------------")

**Output:**



================= RESTART: C:\Users\omkar\Downloads\paandas.py =================

Name ID Branch Year Marks

0 Suma N190706 CSE E-3 99.0

1 Pavani N190001 ECE E-3 NaN

2 Asha NaN MECH E-4 95.0

3 Aswitha N190004 CSE E-2 88.0

4 Ram N191090 CSE E-2 98.0

5 Duswanth N190707 ECE E-2 75.0

6 Naresh NaN CSE E-1 85.0

7 Balaji N190886 CIVIL NaN 60.0

8 Priya N190983 MME E-1 55.0

9 Harshitha N190499 NaN E-2 99.0

Name ID Branch Year Marks

0 Suma N190706 CSE E-3 99.0

1 Pavani N190001 ECE E-3 NaN

2 Asha NaN MECH E-4 95.0

3 Aswitha N190004 CSE E-2 88.0

4 Ram N191090 CSE E-2 98.0

5 Duswanth N190707 ECE E-2 75.0

6 Naresh NaN CSE E-1 85.0

7 Balaji N190886 CIVIL NaN 60.0

8 Priya N190983 MME E-1 55.0

9 Harshitha N190499 NaN E-2 99.0

= RESTART: C:/Users/omkar/Downloads/pandas data saving.py

Series([], dtype: object)

0 10

1 20

2 30

dtype: int64

-------------

{'a': 10, 'b': 20, 'c': 30}

-------------

1 1

2 2

3 3

dtype: int32

-------------

Index([1, 2, 3], dtype='int64')

[1 2 3]

None

(3,)

1

3

12

<bound method Series.memory\_usage of 1 1

2 2

3 3

dtype: int32>

36

False

-------------

0 2

1 4

2 6

3 8

4 10

dtype: int64

0 0

1 0

2 0

3 0

4 0

dtype: int64

0 1

1 4

2 9

3 16

4 25

dtype: int64

0 1.0

1 1.0

2 1.0

3 1.0

4 1.0

dtype: float64

0 0

1 0

2 0

3 0

4 0

dtype: int64

0 1

1 4

2 27

3 256

4 3125

dtype: int64

0 True

1 True

2 True

3 True

4 True

dtype: bool

0 False

1 False

2 False

3 False

4 False

dtype: bool

0 True

1 True

2 True

3 True

4 True

dtype: bool

0 1

1 2

2 3

3 4

4 5

dtype: int64

0 1

1 2

2 3

3 4

4 5

dtype: int64

count 5.000000

mean 3.000000

std 1.581139

min 1.000000

25% 2.000000

50% 3.000000

75% 4.000000

max 5.000000

dtype: float64

0 1

2 3

4 5

dtype: int64

-------------

**ASSIGNMENT-05**

**Working with Numpy Arrays:** Numpy 1d Arrays, Numpy 2d Arrays

**Code:**

import numpy as np

a=np.array([[10,20,30],[40,50,60]])

print(a)

import numpy as np

a = np.array([10])

b = np.array([10,20,30])

c = np.array([[10,20],[30,40]])

d = np.array([[10,20],[30,40],[[50,60],[70,80]]])

print(a.ndim)

print(b.ndim)

print(c.ndim)

print(d.ndim)

import numpy as np

a=[10,20,30]

b=np.asarray(a,dtype=float)

print(b)

a=[[10,20],[30,40]]

b=np.asarray(a,order="c")

print(b)

for i in np.nditer(b):

print(i)

import numpy as np

a=np.zeros((3,4))

print(a)

b=(3,4)

fill\_value=3

c=np.full(b,fill\_value)

print(c)

d=np.random.rand(10)

print(d)

e=np.ones((2,3))

print(e)

f=np.eye(2,dtype= float)

print(f)

print("the size of the array a is : ",np.size(a))

print("the shape of the array c is :",np.shape(c))

print("the datatype of the array d is : ",e.dtype)

import numpy as np

arr = np.arange(10,100,5,int)

print("The array over the given range is ",arr)

a=np.linspace(10,20,5)

print(a)

b=np.linspace(10,20,5,endpoint=False)

print(b)

c=np.logspace(10,20,num=5,endpoint=True)

print(c)

d=np.logspace(10,20,num=3,base=2,endpoint=False)

print(d)

import numpy as np

a=np.array([[10,20,30],[40,50,60]])

print(a[1,2])

print(a[0,2])

arr = np.arange(6)

print("array arr:",arr)

print("sliced element of array:",arr[1:3])

arr=np.arange(12)

arr1=arr.reshape(3,4)

print("Array arr1:\n",arr1)

print("\n")

print("elements of 1st row and 1st column upto last column :\n",arr1[1:,1:4])

import numpy as np

a=np.arange(5)

b=np.copy(a)

print(b)

c=a.view()

print(c)

d=np.array([35,20,70,11,50,33])

print(np.sort(d))

e=np.array([10,20,30,40]).reshape(4,1)

print(e)

a=np.array([[10, 20, 30], [40, 50, 60], [70, 80, 90]])

b=np.array([[11, 21, 31], [42, 52, 62], [73, 83, 93]])

c=np.append(a,b,axis=0)

d=np.concatenate((a,b),axis=0)

print(c)

print(d)

arr=np.array([10,30,40,50])

arr1=np.delete(arr,0)

print(arr1)

np.stack(arrays, axis=1).shape

np.stack(arrays, axis=2).shape

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

y = np.array([3, 4, 5])

np.vstack((x, y))

np.stack(arrays, axis=1).shape

np.stack(arrays, axis=2).shape

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

y = np.array([3, 4, 5])

np.stack((x, y))

np.stack(arrays, axis=1).shape

np.stack(arrays, axis=2).shape

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

y = np.array([3, 4, 5])

np.hstack((x, y))

np.stack(arrays, axis=1).shape

np.stack(arrays, axis=2).shape

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

y = np.array([3, 4, 5])

np.dstack((x, y))

import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6])

newarr = np.array\_split(arr, 3)

print(newarr)

import numpy as np

np.where([[True, False], [True, True]],

[[1, 2], [3, 4]], [[5, 6], [7, 8]])

d=np.array([35,20,70,11,50,33])

print("the sum of the array is:",np.sum(d))

print("the min value in the array is :",np.min(d))

print("the max value in the array is :",np.max(d))

print("the mean of the array is :",np.mean(d))

print("the median of the array is :",np.median(d))

print("the variance of the array is :",np.var(d))

print("the standard deviation of the array is :",np.std(d))

**Output:**

[[10 20 30]

[40 50 60]]

1

1

2

2

[10. 20. 30.]

[[10 20]

[30 40]]

10

20

30

40

[[0. 0. 0. 0.]

[0. 0. 0. 0.]

[0. 0. 0. 0.]]

[[3 3 3 3]

[3 3 3 3]

[3 3 3 3]]

[0.57741241 0.26083806 0.33748906 0.43851002 0.80732267 0.9744915

0.46086947 0.02763557 0.56793034 0.375695 ]

[[1. 1. 1.]

[1. 1. 1.]]

[[1. 0.]

[0. 1.]]

the size of the array a is : 12

the shape of the array c is : (3, 4)

the datatype of the array d is : float64

The array over the given range is [10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95]

[10. 12.5 15. 17.5 20. ]

[10. 12. 14. 16. 18.]

[1.00000000e+10 3.16227766e+12 1.00000000e+15 3.16227766e+17

1.00000000e+20]

[ 1024. 10321.27324074 104031.91534179]

60

30

array arr: [0 1 2 3 4 5]

sliced element of array: [1 2]

Array arr1:

[[ 0 1 2 3]

[ 4 5 6 7]

[ 8 9 10 11]]

elements of 1st row and 1st column upto last column :

[[ 5 6 7]

[ 9 10 11]]

[0 1 2 3 4]

[0 1 2 3 4]

[11 20 33 35 50 70]

[[10]

[20]

[30]

[40]]

[[10 20 30]

[40 50 60]

[70 80 90]

[11 21 31]

[42 52 62]

[73 83 93]]

[[10 20 30]

[40 50 60]

[70 80 90]

[11 21 31]

[42 52 62]

[73 83 93]]

[30 40 50]

array([[2, 3, 4],

[3, 4, 5]])

array([[2, 3, 4],

[3, 4, 5]])

array([2, 3, 4, 3, 4, 5])

array([[[2, 3],

[3, 4],

[4, 5]]])

[array([1, 2]), array([3, 4]), array([5, 6])]

import numpy as np

np.where([[True, False], [True, True]],

[[1, 2], [3, 4]], [[5, 6], [7, 8]])

array([[1, 6],

[3, 4]])

the sum of the array is: 219

the min value in the array is : 11

the max value in the array is : 70

the mean of the array is : 36.5

the median of the array is : 34.0

the variance of the array is : 373.5833333333333

the standard deviation of the array is : 19.328303943526272

**ASSIGNMENT-06**

**Importing Datasets and Cleaning and Preparing the Data :** Importing and Exporting Data in Python,Identify and Handle Missing Values, Data

Formatting, Data Normalization Sets, Binning, Indicator variables

**Code:**

import pandas as pd

import numpy as np

df=pd.read\_excel("sgrades.xlsx")

print(df)

print(df['sname'])#dataframe with single column

print(df[['sname','Total']])#dataframe with multiple column

print(df[['sname','Total'][3:5:2]])#dataframe with multiple column with slicing

for index\_val,row\_val in df.iterrows():

print(index\_val,row\_val)#for loop iteration of rows

print(df.loc[1:6])

print(df.sort\_values("Total",ascending=False))

print(df.sort\_values("sname"))

print(df.sort\_values(["sname","Total"],ascending=[1,0]))

df=pd.read\_excel("sgrades.xlsx")

print(df.dropna())

print(df.fillna("Missing",inplace=True))

df.to\_csv("Text.txt",index=False,sep="\t")

df.loc[5,'sname']

df.loc[0:4,['sname','Total']]

df.loc[0:4,'sname':'Total']

df.iloc[:,[1,4]]

df.iloc[0:4,1:7]

df=pd.read\_csv('Grading of the students in the exam (OR).csv')

print(df)

import pandas as pd

df=pd.read\_excel("pandas.xlsx")

print(df)

print(df.loc[0:4,['Name','ID','Branch']])

print(df.iloc[0:4,0:3])

print(df.dropna())

print(df.fillna("missing"))

print(df.isna())

df2 = df.dropna(how='all').dropna(how='all', axis=1)

print(df2)

import numpy as np

import pandas as pd

df=pd.read\_csv('pandas.csv')

def z\_score\_normalization(data):

mean = np.mean(data)

std = np.std(data)

normalized\_data = (data - mean) / std

return normalized\_data

# Example usage

data = df['marks']

normalized\_data = z\_score\_normalization(data)

print(normalized\_data)

**Output:**

= RESTART: C:/Users/omkar/Downloads/importing and expoting the data.py

sno sname Tel Eng Maths Total

0 1 Asha 10 10 8 28

1 2 Pavani 10 10 10 30

2 3 Rosy 10 10 5 25

3 4 Joshi 5 5 10 20

4 5 John 8 8 8 24

5 6 Peter 7 7 8 22

6 7 Kivi 5 7 7 19

7 8 Sam 2 10 10 22

8 9 Ram 4 5 6 15

9 10 Lucky 6 6 6 18

0 Asha

1 Pavani

2 Rosy

3 Joshi

4 John

5 Peter

6 Kivi

7 Sam

8 Ram

9 Lucky

Name: sname, dtype: object

sname Total

0 Asha 28

1 Pavani 30

2 Rosy 25

3 Joshi 20

4 John 24

5 Peter 22

6 Kivi 19

7 Sam 22

8 Ram 15

9 Lucky 18

Empty DataFrame

Columns: []

Index: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

0 sno 1

sname Asha

Tel 10

Eng 10

Maths 8

Total 28

Name: 0, dtype: object

1 sno 2

sname Pavani

Tel 10

Eng 10

Maths 10

Total 30

Name: 1, dtype: object

2 sno 3

sname Rosy

Tel 10

Eng 10

Maths 5

Total 25

Name: 2, dtype: object

3 sno 4

sname Joshi

Tel 5

Eng 5

Maths 10

Total 20

Name: 3, dtype: object

4 sno 5

sname John

Tel 8

Eng 8

Maths 8

Total 24

Name: 4, dtype: object

5 sno 6

sname Peter

Tel 7

Eng 7

Maths 8

Total 22

Name: 5, dtype: object

6 sno 7

sname Kivi

Tel 5

Eng 7

Maths 7

Total 19

Name: 6, dtype: object

7 sno 8

sname Sam

Tel 2

Eng 10

Maths 10

Total 22

Name: 7, dtype: object

8 sno 9

sname Ram

Tel 4

Eng 5

Maths 6

Total 15

Name: 8, dtype: object

9 sno 10

sname Lucky

Tel 6

Eng 6

Maths 6

Total 18

Name: 9, dtype: object

sno sname Tel Eng Maths Total

1 2 Pavani 10 10 10 30

2 3 Rosy 10 10 5 25

3 4 Joshi 5 5 10 20

4 5 John 8 8 8 24

5 6 Peter 7 7 8 22

6 7 Kivi 5 7 7 19

sno sname Tel Eng Maths Total

1 2 Pavani 10 10 10 30

0 1 Asha 10 10 8 28

2 3 Rosy 10 10 5 25

4 5 John 8 8 8 24

5 6 Peter 7 7 8 22

7 8 Sam 2 10 10 22

3 4 Joshi 5 5 10 20

6 7 Kivi 5 7 7 19

9 10 Lucky 6 6 6 18

8 9 Ram 4 5 6 15

sno sname Tel Eng Maths Total

0 1 Asha 10 10 8 28

4 5 John 8 8 8 24

3 4 Joshi 5 5 10 20

6 7 Kivi 5 7 7 19

9 10 Lucky 6 6 6 18

1 2 Pavani 10 10 10 30

5 6 Peter 7 7 8 22

8 9 Ram 4 5 6 15

2 3 Rosy 10 10 5 25

7 8 Sam 2 10 10 22

sno sname Tel Eng Maths Total

0 1 Asha 10 10 8 28

4 5 John 8 8 8 24

3 4 Joshi 5 5 10 20

6 7 Kivi 5 7 7 19

9 10 Lucky 6 6 6 18

1 2 Pavani 10 10 10 30

5 6 Peter 7 7 8 22

8 9 Ram 4 5 6 15

2 3 Rosy 10 10 5 25

7 8 Sam 2 10 10 22

sno sname Tel Eng Maths Total

0 1 Asha 10 10 8 28

1 2 Pavani 10 10 10 30

2 3 Rosy 10 10 5 25

3 4 Joshi 5 5 10 20

4 5 John 8 8 8 24

5 6 Peter 7 7 8 22

6 7 Kivi 5 7 7 19

7 8 Sam 2 10 10 22

8 9 Ram 4 5 6 15

9 10 Lucky 6 6 6 18

None

Name ID Branch Address

0 Suma N190699 CSE NaN

1 Malathi N190436 CSE Srikaklm

2 Pavani NaN ECE Guntur

3 Asha N190900 CSE NaN

4 Shakku N190684 CSE Chittoor

5 Sufiya N190514 NaN Anantapur

6 Lokesh N190321 MECH NaN

7 Karthik N190782 EEE NaN

8 Priya N190632 NaN Vyzag

Name ID Branch

0 Suma N190699 CSE

1 Malathi N190436 CSE

2 Pavani NaN ECE

3 Asha N190900 CSE

4 Shakku N190684 CSE

Name ID Branch

0 Suma N190699 CSE

1 Malathi N190436 CSE

2 Pavani NaN ECE

3 Asha N190900 CSE

Name ID Branch Address

1 Malathi N190436 CSE Srikaklm

4 Shakku N190684 CSE Chittoor

Name ID Branch Address

0 Suma N190699 CSE missing

1 Malathi N190436 CSE Srikaklm

2 Pavani missing ECE Guntur

3 Asha N190900 CSE missing

4 Shakku N190684 CSE Chittoor

5 Sufiya N190514 missing Anantapur

6 Lokesh N190321 MECH missing

7 Karthik N190782 EEE missing

8 Priya N190632 missing Vyzag

Name ID Branch Address

0 False False False True

1 False False False False

2 False True False False

3 False False False True

4 False False False False

5 False False True False

6 False False False True

7 False False False True

8 False False True False

Name ID Branch Address

0 Suma N190699 CSE NaN

1 Malathi N190436 CSE Srikaklm

2 Pavani NaN ECE Guntur

3 Asha N190900 CSE NaN

4 Shakku N190684 CSE Chittoor

5 Sufiya N190514 NaN Anantapur

6 Lokesh N190321 MECH NaN

7 Karthik N190782 EEE NaN

8 Priya N190632 NaN Vyzag

0 -1.517849

1 0.141002

2 -0.688423

3 1.219256

4 0.846014

Name: marks, dtype: float64

**ASSIGNMENT-07**

**Model Development:** Simple and Multiple Linear Regression, Model

EvaluationUsingVisualization, Polynomial Regression and Pipelines, R-squared and MSE for InSample Evaluation, Prediction and Decision Making

**Code:**

import pandas as pd

df=pd.read\_csv("data.csv")

df.head()

data\_=df.loc[:,['Weight','CO2']]

print(data\_.head(10))

#showing the data in matplotlib

#to use we need to first install matplotlib

import matplotlib.pyplot as plt

df.plot(x='Weight',y='CO2',style='o')

plt.xlabel('Weight')

plt.ylabel('CO2')

plt.show()

#dividing the variables into dependent and independent

X=pd.DataFrame(df['Weight'])

y=pd.DataFrame(df['CO2'])

#Split the data into train and test sets

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

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.2,random\_state=1)

#knowing the shapes of the test and train

print(X\_train.shape)

print(X\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

#train the algorithm

from sklearn.linear\_model import LinearRegression

regressor=LinearRegression()

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

#retriving the intercept

print(regressor.intercept\_)

#retriving the slope

print(regressor.coef\_)

#predecting the test results

y\_pred = regressor.predict(X\_test)

y\_test

print(y\_pred)

print(y\_test)

#evaluting the algorithm

from sklearn import metrics

import numpy as np

print('Mean Absolute Error:',metrics.mean\_absolute\_error(y\_test,y\_pred))

print('Mean Squared Error:',metrics.mean\_squared\_error(y\_test,y\_pred))

print('Root Mean Squared Error:',np.sqrt(metrics.mean\_squared\_error(y\_test,y\_pred)))

#plot for the train set

plt.scatter(X\_train, y\_train, color='red') # plotting the observation line

plt.plot(X\_train, regressor.predict(X\_train), color='blue') # plotting the regression line

plt.title("Weight vs CO2 (Training set)") # stating the title of the graph

plt.xlabel("Weight") # adding the name of x-axis

plt.ylabel("CO2") # adding the name of y-axis

plt.show() # specifies end of graph

#plot for the test set

plt.scatter(X\_test, y\_test, color='red')

plt.plot(X\_train, regressor.predict(X\_train), color='blue') # plotting the regression line

plt.title("Weight vs CO2 (Testing set)")

plt.xlabel("Weight")

plt.ylabel("CO2")

plt.show()

#importing pandas

import pandas as pd

#importing data set

df=pd.read\_csv("data.csv")

#making list of independent variales as x and dependent variable as y

X = df[['Weight', 'Volume']]

y = df['CO2']

#to import this sklearn pip install -U scikit-learn

from sklearn import linear\_model

regr = linear\_model.LinearRegression()

regr.fit(X, y)

predictedCO2 = regr.predict([[2300, 1300]])

print(predictedCO2)

print(regr.coef\_)

predictedCO2 = regr.predict([[3300, 1300]])

print(predictedCO2)

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

df=pd.read\_csv('data.csv')

x=df['Weight']

y=df['CO2']

mymodel=np.poly1d(np.polyfit(x, y, 3))

myline=np.linspace(1,30,100)

plt.scatter(x,y)

plt.plot(myline,mymodel(myline))

plt.show()

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_diabetes

from sklearn.linear\_model import Ridge

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

from sklearn.metrics import mean\_squared\_error

# Load the dataset

diabetes = load\_diabetes()

# Separate the features and target variable

X = diabetes.data

y = diabetes.target

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize and fit the Ridge regression model

ridge = Ridge(alpha=1.0) # You can adjust the regularization strength with the 'alpha' parameter

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

# Predict on the test set

y\_pred = ridge.predict(X\_test)

# Calculate coefficients and intercept

coefficients = ridge.coef\_

intercept = ridge.intercept\_

# Print the coefficients and intercept

print("Coefficients:", coefficients)

print("Intercept:", intercept)

# Calculate mean squared error

mse = mean\_squared\_error(y\_test, y\_pred)

print("Mean Squared Error:", mse)

# Plot the predicted values against the true values

plt.scatter(y\_test, y\_pred)

plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2)

plt.xlabel('True Values')

plt.ylabel('Predicted Values')

plt.title('Ridge Regression - True vs Predicted')

plt.show()

**Output:**

Weight CO2

0 790 99

1 1160 95

2 929 95

3 865 90

4 1140 105

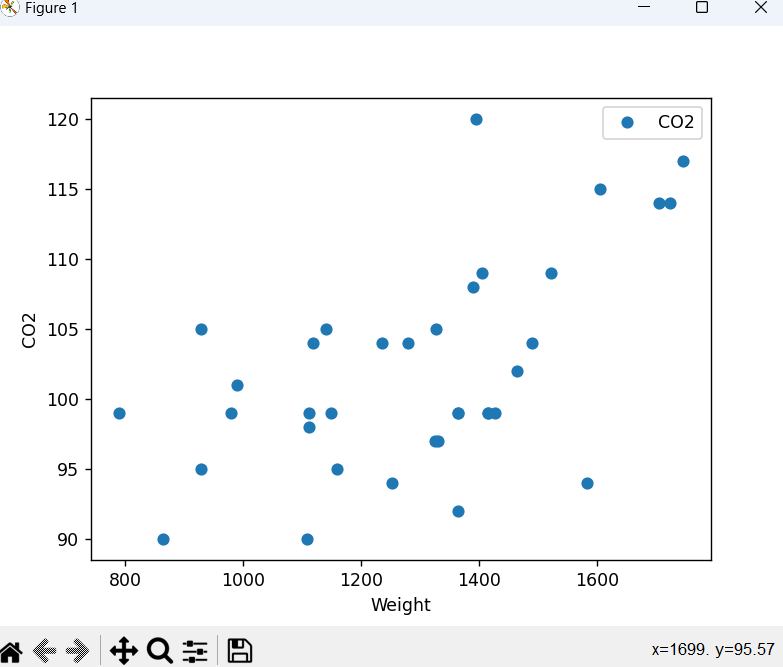
5 929 105

6 1109 90

7 1365 92

8 1112 98

9 1150 99



(28, 1)

(8, 1)

(28, 1)

(8, 1)

[83.33027919]

[[0.01428958]]

[[106.26505488]

[103.40713891]

[105.09330933]

[ 95.69076578]

[102.30684126]

[101.62094142]

[103.73579924]

[103.5500347 ]]

CO2

30 115

34 109

28 109

3 90

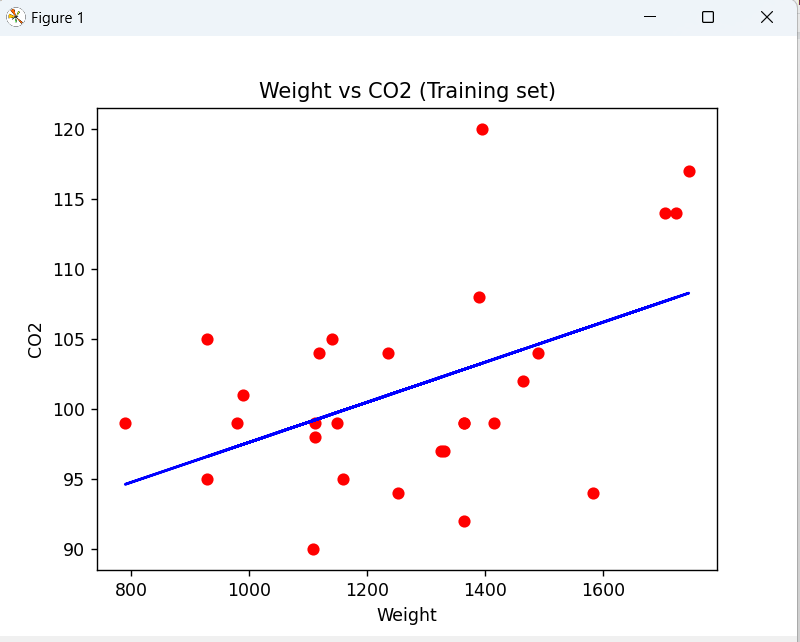
19 105

17 104

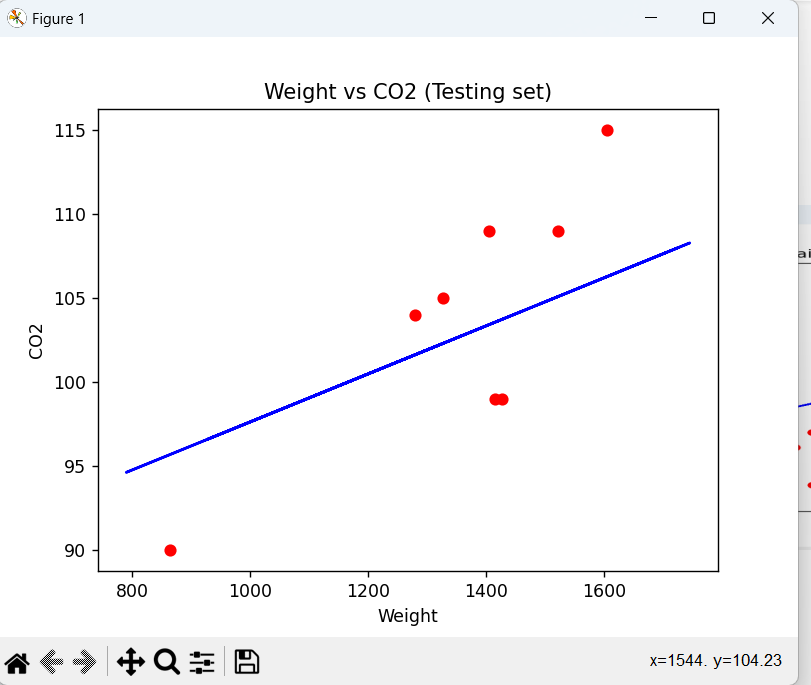
21 99

23 99

Mean Absolute Error: 4.785414241420883

Mean Squared Error: 26.40875532851579

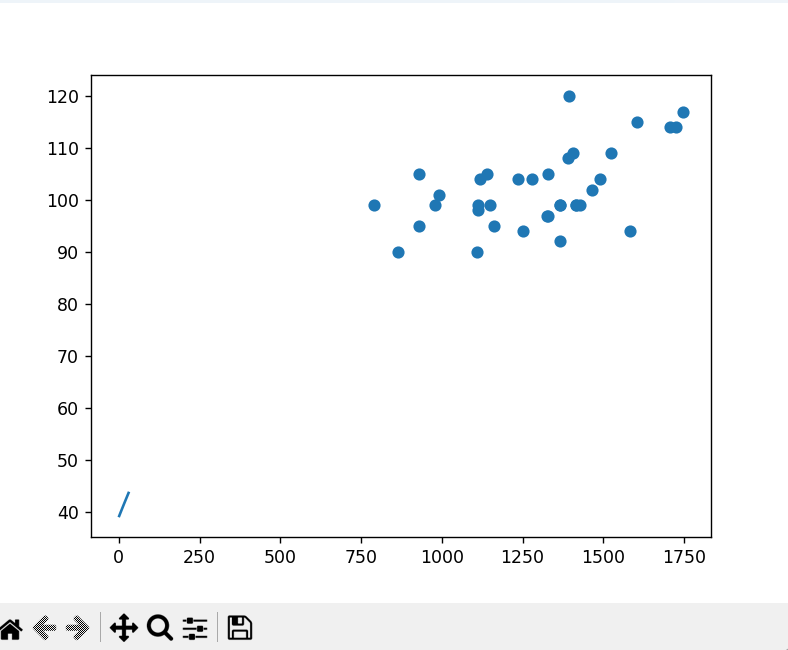
Root Mean Squared Error: 5.138944962588702



[107.2087328]

[0.00755095 0.00780526]

[114.75968007]

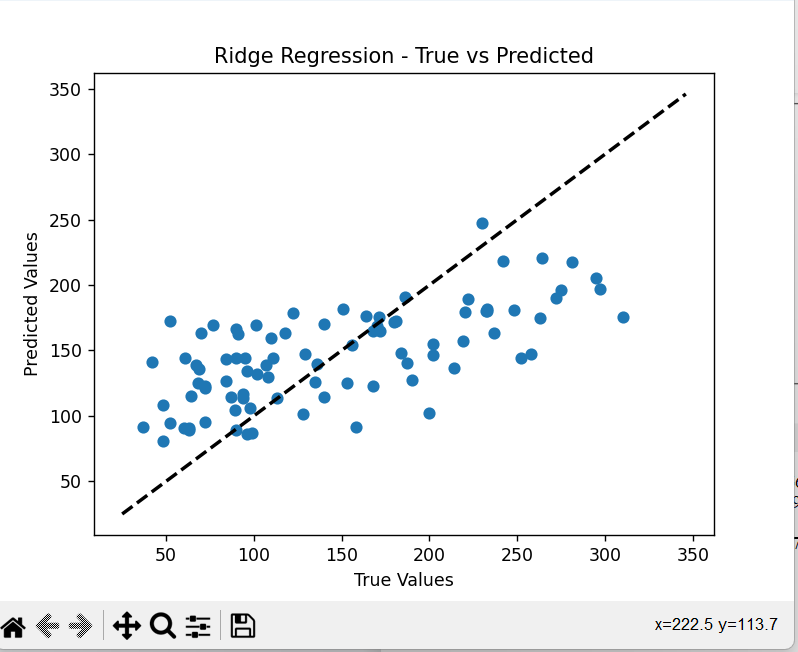


Coefficients: [ 45.36737726 -76.66608563 291.33883165 198.99581745 -0.53030959

-28.57704987 -144.51190505 119.26006559 230.22160832 112.14983004]

Intercept: 152.241675211113

Mean Squared Error: 3077.41593882723



**ASSIGNMENT-08**

**Model Evaluation:** Model Evaluation, Over-fitting, Under-fitting and Model Selection, Ridge Regression, Grid Search, Model Refinement

**Code:**

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Generate some sample data

np.random.seed(42)

X = np.random.rand(100, 1) \* 10

y = 2 \* X + np.random.randn(100, 1)

# Split the data into training and validation sets

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Fit a linear regression model on the training data

model = LinearRegression()

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

# Make predictions on training and validation data

y\_train\_pred = model.predict(X\_train)

y\_val\_pred = model.predict(X\_val)

# Calculate mean squared errors

train\_error = mean\_squared\_error(y\_train, y\_train\_pred)

val\_error = mean\_squared\_error(y\_val, y\_val\_pred)

# Plot the learning curves

plt.plot(X\_train, y\_train, 'bo', label='Training data')

plt.plot(X\_val, y\_val, 'ro', label='Validation data')

plt.plot(X\_train, y\_train\_pred, 'g-', label='Training predictions')

plt.plot(X\_val, y\_val\_pred, 'm-', label='Validation predictions')

plt.legend()

plt.xlabel('X')

plt.ylabel('y')

plt.title('Linear Regression')

plt.show()

print('Training MSE:', train\_error)

print('Validation MSE:', val\_error)

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Generate some sample data

np.random.seed(42)

X = np.random.rand(100, 1) \* 10

y = 2 \* X + np.random.randn(100, 1)

# Split the data into training and validation sets

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Fit a linear regression model on a subset of the training data

model = LinearRegression()

model.fit(X\_train[:10], y\_train[:10])

# Make predictions on training and validation data

y\_train\_pred = model.predict(X\_train)

y\_val\_pred = model.predict(X\_val)

# Calculate mean squared errors

train\_error = mean\_squared\_error(y\_train, y\_train\_pred)

val\_error = mean\_squared\_error(y\_val, y\_val\_pred)

# Plot the learning curves

plt.plot(X\_train, y\_train, 'bo', label='Training data')

plt.plot(X\_val, y\_val, 'ro', label='Validation data')

plt.plot(X\_train, y\_train\_pred, 'g-', label='Training predictions')

plt.plot(X\_val, y\_val\_pred, 'm-', label='Validation predictions')

plt.legend()

plt.xlabel('X')

plt.ylabel('y')

plt.title('Linear Regression')

plt.show()

print('Training MSE:', train\_error)

print('Validation MSE:', val\_error)

import numpy as np

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error

# Generate some sample data

np.random.seed(42)

X = np.random.rand(100, 1) \* 10

y = 2 \* X + np.random.randn(100, 1)

# Define the models

models = [

('Linear Regression', LinearRegression()),

('Decision Tree', DecisionTreeRegressor()),

('Random Forest', RandomForestRegressor())

]

# Evaluate each model using cross-validation

for model\_name, model in models:

scores = cross\_val\_score(model, X, y, scoring='neg\_mean\_squared\_error', cv=5)

rmse\_scores = np.sqrt(-scores)

avg\_rmse = np.mean(rmse\_scores)

print(model\_name)

print('RMSE scores:', rmse\_scores)

print('Average RMSE:', avg\_rmse)

print('---')

import numpy as np

from sklearn.datasets import load\_iris

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

from sklearn.svm import SVC

# Load the iris dataset

iris = load\_iris()

X, y = iris.data, iris.target

# Split the data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Define the parameter grid for grid search

param\_grid = {

'C': [0.1, 1, 10, 100], # Regularization parameter

'kernel': ['linear', 'rbf', 'poly'] # Kernel function

}

# Create a SVM classifier

svm = SVC()

# Create the GridSearchCV object

grid\_search = GridSearchCV(estimator=svm, param\_grid=param\_grid, cv=5, scoring='accuracy')

# Perform grid search on the training data

grid\_search.fit(X\_train, y\_train)

# Print the best hyperparameters and the corresponding accuracy

print("Best hyperparameters:", grid\_search.best\_params\_)

print("Best accuracy:", grid\_search.best\_score\_)

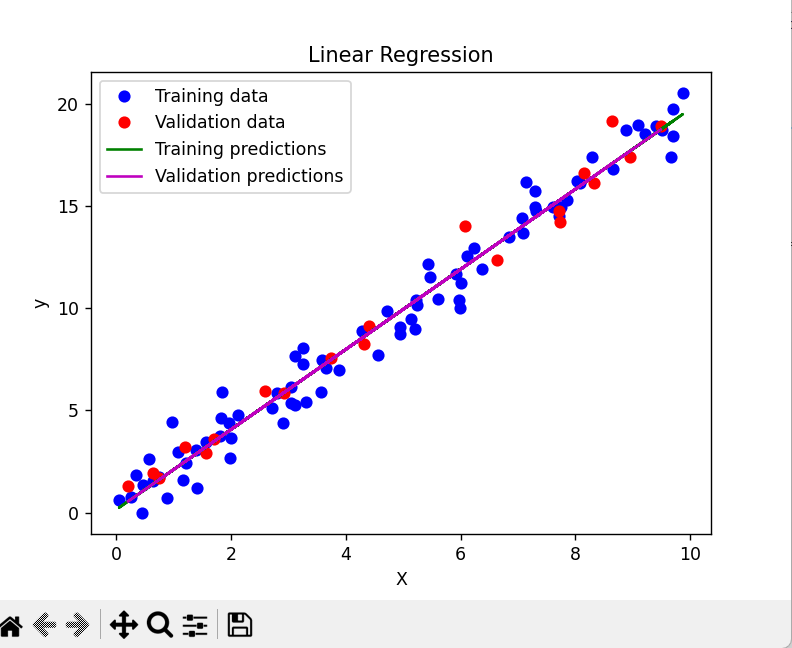
# Evaluate the model on the test data using the best hyperparameters

best\_model = grid\_search.best\_estimator\_

test\_accuracy = best\_model.score(X\_test, y\_test)

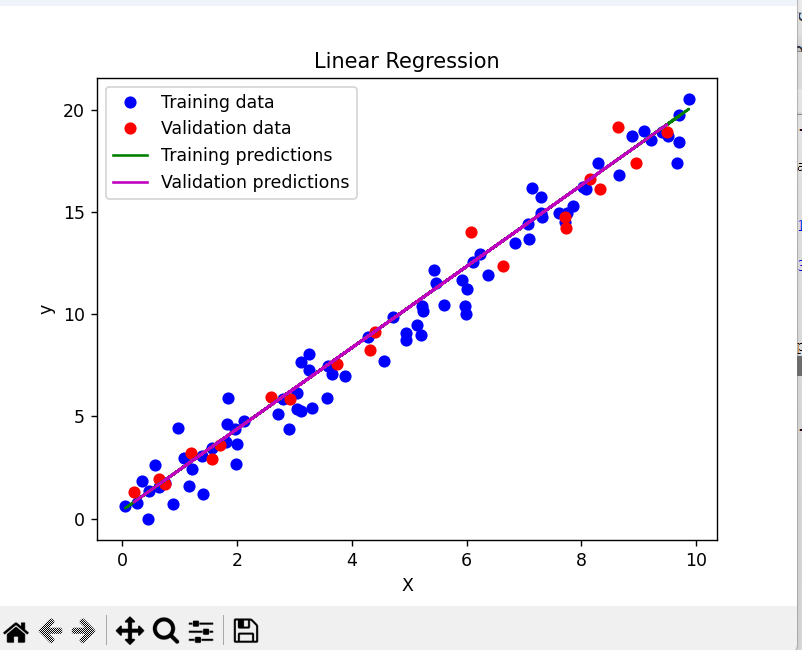
print("Test accuracy with best hyperparameters:", test\_accuracy)

**Output:**

****

Training MSE: 0.8476788564209707

Validation MSE: 0.653699513717003



Training MSE: 1.009101708602413

Validation MSE: 0.6575637436381261

Linear Regression

RMSE scores: [0.77953381 0.89086877 1.0326569 0.90782046 0.98769618]

Average RMSE: 0.919715223669806

---

Decision Tree

RMSE scores: [1.08188241 1.45774641 1.36210368 0.91080956 1.36022072]

Average RMSE: 1.2345525556529042

---

Random Forest

RMSE scores: [0.95060216 1.19257089 1.16866416 0.9016125 1.19856087]

Average RMSE: 1.0824021156248491

---

Best hyperparameters: {'C': 1, 'kernel': 'linear'}

Best accuracy: 0.9583333333333334

Test accuracy with best hyperparameters: 1.0

**ASSIGNMENT-09**

**Introduction to Visualization Tools:** Introduction to Data Visualization,Introduction to Matplotlib, Basic Plotting with Matplotlib,Dataset on Immigration to Canada, Line Plots

**Code:**

import pandas as pd

import matplotlib.pyplot as plt

# Read the CSV file into a pandas DataFrame

data = pd.read\_csv('data.csv')

# Extract the x and y data from the DataFrame

x = data['Volume']

y = data['CO2']

# Plot the data

plt.plot(x, y)

plt.xlabel('Volume')

plt.ylabel('CO2')

plt.title('Data Plot')

plt.show()

import pandas as pd

from matplotlib import pyplot as plt

df=pd.read\_csv("Salary\_Data.csv")

print(df)

x=df.loc[:,"super"]

y=df.loc[:,"salary"]

plt.plot(x,y,linewidth="2",color="red")

plt.xlabel(" Years of experience ")

plt.ylabel(" Salary ")

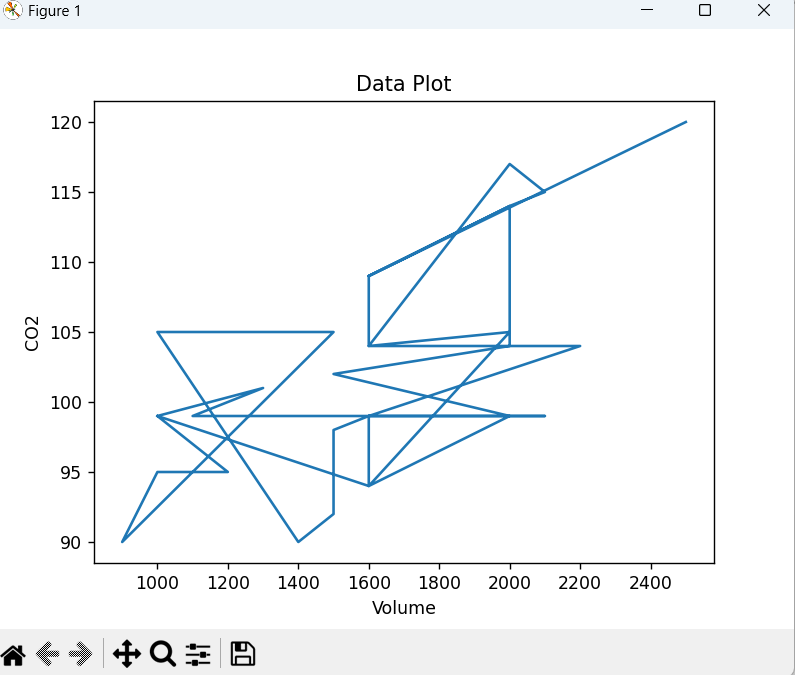
plt.title("Salary based on years of Experience")

plt.grid()

plt.tight\_layout()

plt.show()

**Output:**

****

super salary

0 1.1 35002

1 1.3 39343

2 1.5 46205

3 2.0 37731

4 2.2 43525

5 2.9 39891

6 3.0 56642

7 3.2 60150

8 3.2 54445

9 3.7 64445

10 3.9 57189

11 4.0 63218

12 4.0 55794

13 4.1 56957



**ASSIGNMENT-10**

**Basic Visualization Tools:** Area Plots,Histograms,Bar Charts

**Code:**

from matplotlib import pyplot as plt

wd=[0,1,2,3,4,5,6,7]

emp1=[0,1,2,5,6,7,7,8]

emp2=[0,2,3,4,5,6,6,7]

emp3=[0,1,3,4,5,5,6,6]

labels=['emp1','emp2','emp3']

colors=['green','yellow','blue']

plt.stackplot(wd,emp1,emp2,emp3 ,labels=labels,colors=colors)

plt.legend(loc='upper left')

plt.title('Working hours of employes')

plt.tight\_layout()

plt.show()

import pandas as pd

from matplotlib import pyplot as plt

plt.style.use('fivethirtyeight')

df=pd.read\_csv("Salary\_Data.csv")

print(df)

x=df["super"]

y=df["salary"]

plt.hist(y,bins=4,edgecolor='black',color='blue')

plt.title('Salary of employes')

plt.ylabel('Year of experience')

plt.xlabel('Salary')

plt.tight\_layout()

plt.show()

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

plt.xkcd()

data = {'C':20, 'C++':15, 'Java':30,

'Python':35,'Ruby':12,'R':10}

courses = list(data.keys())

values = list(data.values())

plt.bar( courses,values,color="yellow",width=0.5)

plt.xlabel(" Courses offered ")

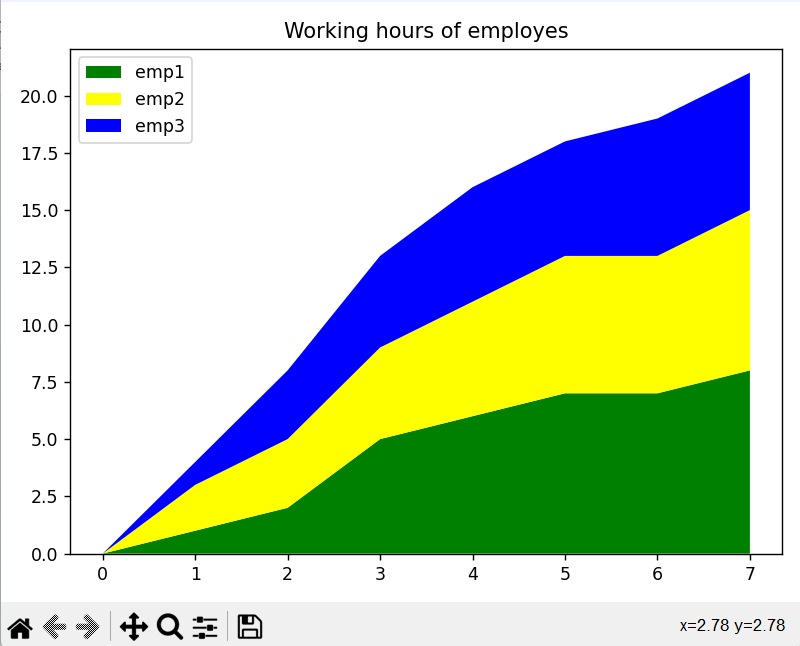
plt.ylabel(" No.of students ")

plt.title("The no.of Students enrolled in different courses")

plt.tight\_layout()

plt.show()

**Output:**

****

super salary

0 1.1 35002

1 1.3 39343

2 1.5 46205

3 2.0 37731

4 2.2 43525

5 2.9 39891

6 3.0 56642

7 3.2 60150

8 3.2 54445

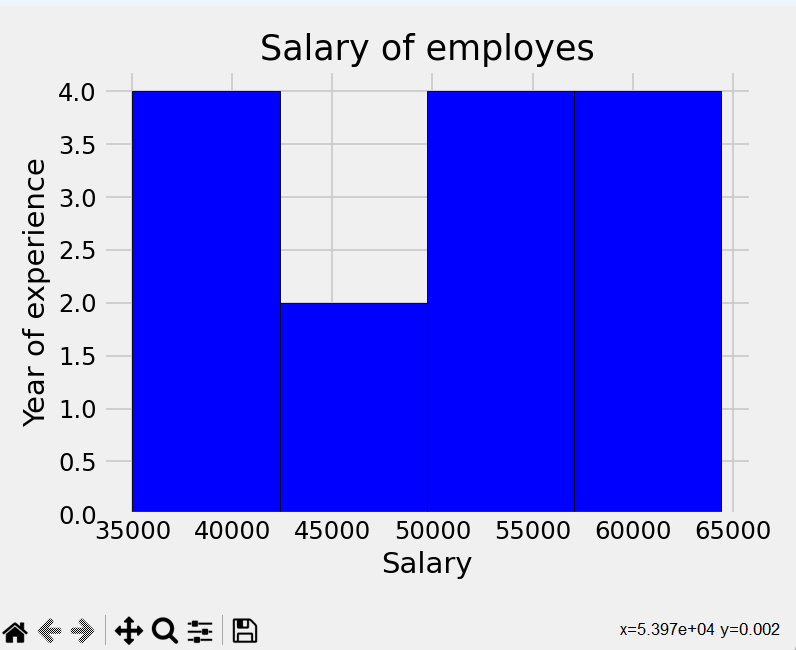
9 3.7 64445

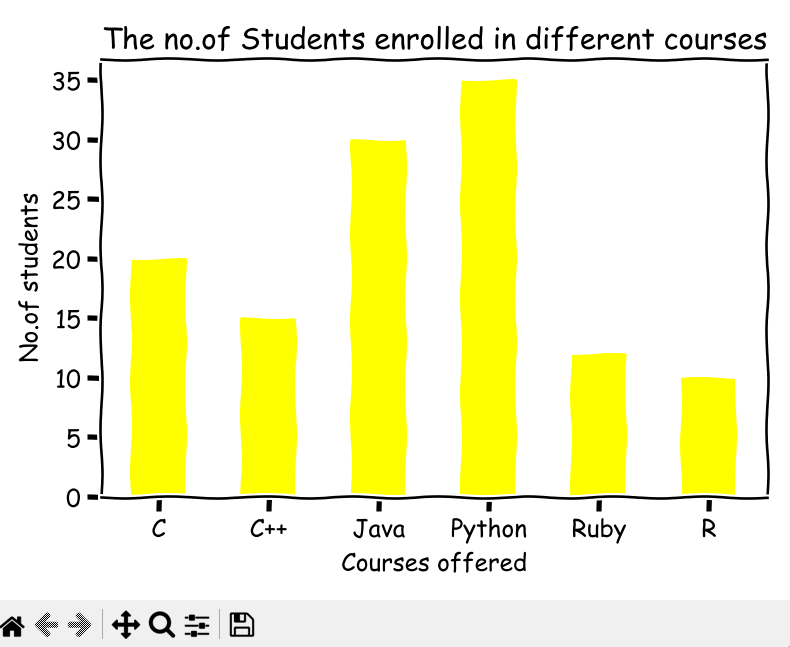
10 3.9 57189

11 4.0 63218

12 4.0 55794

13 4.1 56957

****

****

**ASSIGNMENT-11**

**Specialized Visualization Tools:** Pie Charts, Box Plots, Scatter Plots, Bubble Plots

**Code:**

from matplotlib import pyplot as plt

import pandas as pd

import numpy as np

df=pd.read\_csv('courses.csv')

print(df)

course=df.loc[:,"courses"]

student=df.loc[:,"students"]

explode=[0,0,0.5,0,0]

plt.pie(student,labels=course,wedgeprops={'edgecolor':'white'},shadow=True,explode=explode,startangle=90,autopct='%1.1f%%')

plt.title("Students enrolled in courses")

plt.tight\_layout()

plt.show()

import matplotlib.pyplot as plt

import numpy as np

# Generate some sample data

np.random.seed(42)

data = np.random.normal(0, 1, 100)

# Create a box plot

plt.boxplot(data)

# Add labels and title

plt.xlabel('Data')

plt.ylabel('Values')

plt.title('Box Plot')

# Display the plot

plt.show()

from matplotlib import pyplot as plt

import pandas as pd

plt.style.use('seaborn')

df=pd.read\_csv('Salary\_Data.csv')

print(df)

x=df.loc[:,"super"]

y=df.loc[:,"salary"]

plt.scatter(x,y,s=70,c='red',cmap='greens',edgecolor='black',linewidth=1,alpha=0.75)

plt.title(' salary based on experience')

plt.xlabel("Years of experience")

plt.ylabel("Salary")

plt.tight\_layout()

plt.show()

import matplotlib.pyplot as plt

import numpy as np

# Generate some sample data

np.random.seed(42)

x = np.random.rand(50)

y = np.random.rand(50)

sizes = np.random.randint(10, 100, 50) # Random sizes for the bubbles

# Create a bubble plot

plt.scatter(x, y, s=sizes, alpha=0.5)

# Add labels and title

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Bubble Plot')

# Display the plot

plt.show()

**Output:**

courses students

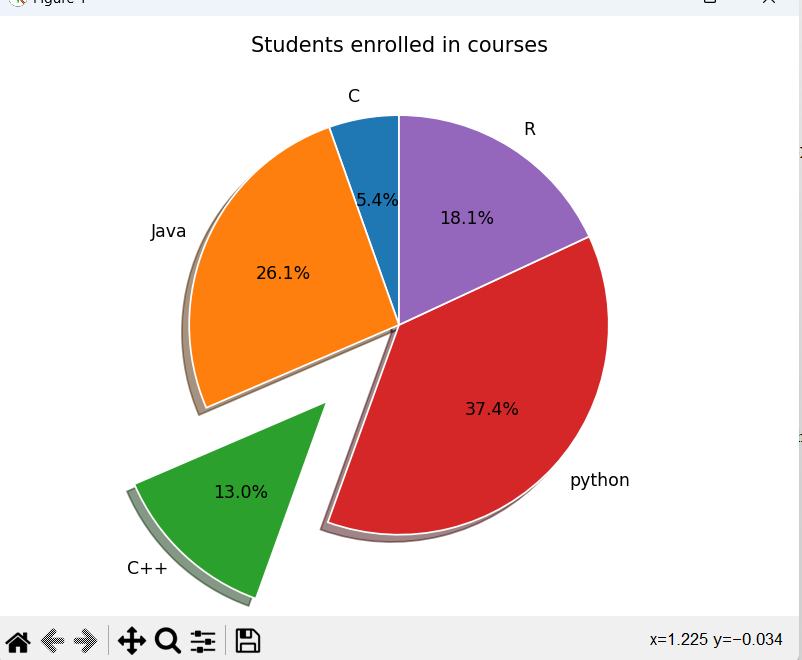
0 C 10.5

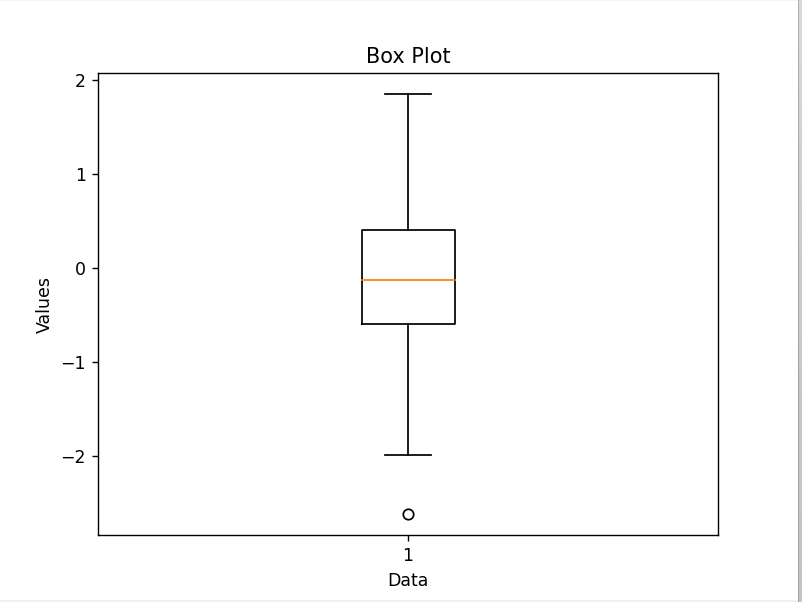
1 Java 50.6

2 C++ 25.3

3 python 72.7

4 R 35.1



****

super salary

0 1.1 35002

1 1.3 39343

2 1.5 46205

3 2.0 37731

4 2.2 43525

5 2.9 39891

6 3.0 56642

7 3.2 60150

8 3.2 54445

9 3.7 64445

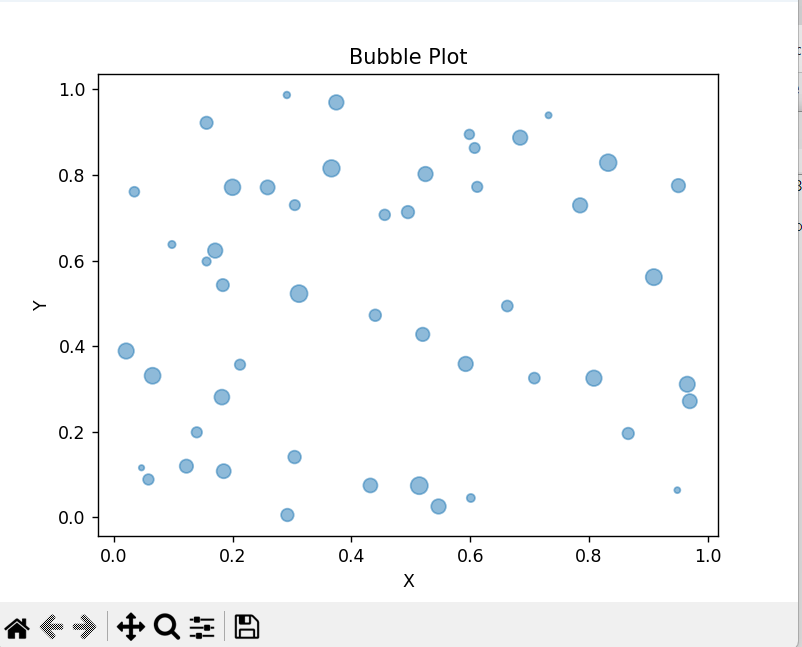
10 3.9 57189

11 4.0 63218

12 4.0 55794

13 4.1 56957





**ASSIGNMENT-12**

**Advanced Visualization Tools:** Waffle Charts, Word Clouds,Seaborn and Regression Plots

**Code:**

import pandas as pd

import matplotlib.pyplot as plt

from pywaffle import Waffle

data={'phone':['xiamoi','samsung','Apple','Nokia','Realme'],'stock':[44,12,8,5,3]}

df=pd.DataFrame(data)

fig=plt.figure(FigureClass=Waffle,rows=5,values=df.stock,labels=list(df.phone))

plt.show()

from wordcloud import WordCloud

import matplotlib.pyplot as plt

text="Hello"

wc=WordCloud().generate(text)

plt.imshow(wc)

plt.axis("off")

plt.show()

import seaborn as sns

sns.set(style="ticks")

df=sns.load\_dataset("anscombe")

sns.lmplot(x="x",y="y",data=df)

import seaborn as sns

df=sns.load\_dataset("penguins")

sns.barplot(data=df,x="island",y="body\_mass\_g",hue="sex")

import numpy as np

import seaborn as sns

sns.set(style="white")

rs=np.random.RandomState(10)

d=rs.normal(size=100)

sns.histplot(d,kde=True,color="m")

import numpy as np

import seaborn as sns

df=sns.load\_dataset("titanic")

sns.boxplot(data=df,x="age",y="class")

import folium

import pandas as pd

m=folium.Map(location=[40,-95],zoom\_start=4)

m.save('my\_map.html')

import folium

m=folium.Map(location=[20,0],tiles="OpenStreetMap",zoom\_start=2)

import pandas as pd

data=pd.DataFrame({

'lon':[-58,2,145,30.32,-4.03,-73.57,36.82,-38.5],

'lat':[-34,49,-38,59.93,5.33,45.52,-1.29,-12.97],

'name':['Buenos Aires','Paris','melbourne','St petersbourg','Abidjan','Montreal','Nairobi','Salvador'],

'value':[10,12,40,70,23,43,100,43]

},dtype=str)

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

folium.Marker(

location=[data.iloc[i]['lat'],data.iloc[i]['lon']],

popup=data.iloc[i]['name'],

  ).add\_to(m)

M

**Output:**

****

