PYTHON AND MACHINE LEARNING(BASICS)

A CERIFICATION COURSE CONDUCTED

BY

THE SURE TUST

Skill Upgradation for Rural-Youth Empowerment

https://suretrustforruralyouth.com

Course Training Attended

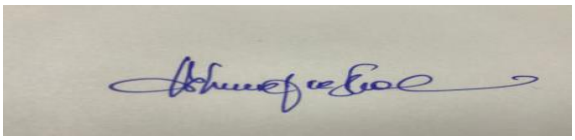
By

Monikaganga Karri

OCTOBER 2022-FEBRUARY2023

DECLARATION

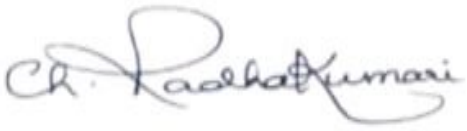
This is to certify that Ms. Monikaganga karri has successfully completed the four months training given in "Python and Machine Learning - Basic" by SURE Trust during the period from October 2022-February 2023.



Mr. Bhuvanesh chellapandian

Trainer

SURE TRUST



Prof.Ch. Radha Kumari

Executive Director&Founder, SURE Trust

Mrs. Vandana Nagesh

Director & Co-Founder, SURE Trust

TABLE OF CONTENTS

1.ABOUT THE SURE TRUST

Introduction to the SURE Trust

The SURE TRUST is born to enhance the employability of educated unemployed rural youth. It is observed that there is a wide gap between the skills acquired by students from the academic institutions and the skills required by the industry to employ them. Employability enhancement is done through giving one on one training in emerging technologies, completely through online mode. The mission of the SURE TRUST is to bridge the gap between the skills acquired and the skills required by training them in the most emerging technologies such as Artificial Intelligence (AI), Python Program, Machine Learning (ML), Deep Learning (DL), Data Science & Data Analytics, Block chain Technology, Robotic Process Automation (RPA), and Project Management and twenty other essential and high in demand Courses, that will enhance their employability. After completion of four months training in the course, the trainees will get live projects from industries as internship activity to get experience in applying to real time situation what they have learnt during the course. These projects will give them hands on experience which is much sought after by the prospective industry employing them. Currently students from all over India are enrolling for various

courses offered by the SURE TRUST. The SURE TRUST offers every course free of cost with no financial burden of any kind to students. This initiative is purely a serviceoriented one aiming to guide the rural youth who are educated but unemployed due to lack of up gradation in their skill sets. The birth of SURE TRUST is a God given boon to rural youth who could reach great heights either in employment or in entrepreneurship once they receive the training offered followed by the company internship. Many companies are coming forward to join their hands with us by offering internship projects to hand hold and lead the rural youth in their career settlement

Vision of THE SURE TRUST

The vision of the SURE TRUST is to enhance the employability of educated unemployed youth, particularly living in rural areas, through skill up gradation, with no cost to the students.

Mission of the SURE TRUST

The mission is to bridge the gap between the skills acquired in the academic institutions and the skills required in industries as a pre-condition for employment.

Functioning of the SURE TRUST

There are three dedicated, committed, and hard-working women on the board of management of the SURE TRUST who will look into the various administrative and other matters relating to the enrolment of students, organizing trainers, entering into agreements with companies for getting live projects to students as internship programs, and so on. All the three women on the board are all the alumni from Sri Sathya Sai Institute of Higher Learning, Ananthapuram Campus, deemed to be a University. The women board is supported by five eminent

advisories who are from different walks of life and have made outstanding mark in career in their respective fields. For more details about SURE TRUST please visit the website https://suretrustforruralyouth.com

2.COURSE CONTENT

The SURE TRUST conducts a four months training for every course on a uniform basis. A session spanning across one to one & half hour is taken by the trainers for every major course. Sessions are conducted to complete the predesigned course structure within the fixed time period. Course content is designed to suit the current requirement of the industry and validated by the industry experts. The course content of all these courses is so dynamic that any changed condition noticed in the industry will automatically get reflected in the content of the respective course. As the course content is dynamic, the Following is the course content of the current course in Data Science and Data Analytics:

PYTHON & MACHINE LEARNING COURSE(BASICS)

OBJECTIVE:

The objective of this course is to develop the skills required for Python and Machine Learning Technology basic with use of python to analyze the data, create visualizations, Developing & evaluating models and get introduced to complex computations.

Course Content:

1. Python for Data Science and Machine Learning
    - Python Basics
2. Data Analysis

- Pandas Library
- Numpy Library

3. Data Visualization
    - Matplotlib library
    - Seaborn Library
4. Exploratory Data Analysis
    - Data Collection
    - Data Cleaning
    - Univariate Analysis
    - Bivariate Analysis
    - Multivariate Analysis
    - Data transformation
5. Supervised Learning
    - Regression
    - Linear regression
    - Logistic Regression
    - Classification
    - Naive Bayes
    - Support vector machine
    - KNN algorithm
    - Random Forest

6.Unsupervised Learning:

- K-means

3.COURSE CONDUCTION

a) Process for the conduct of all the courses are fixed by the SURE TRUST which are

- Uniformly followed across the courses.
- Mode of Training --- Online.
- Period of Training --- Four months
- Sessions per week --- 3 to 6
- Length of the session --- 1 to 2 hours
- Tests/ Quiz to be taken --- 2 per month
- Assignments --- 2 per month
- Last 15 days --- Final practice and preparing the course report

b) Guidelines for the Student

Students who are enrolling for the courses under SURE TRUST are strictly required to follow the following guidelines set for them. Guidelines for students to become eligible for certificate at the end of the Course.

Minimum Attendance

Every student should maintain a minimum attendance of 85% in the entire course duration to eligible for getting the certificate.

Quiz

Since the objective of the certification program is to turn out well qualified students from the respective courses, quizzes are to be taken for each course to ensure that the students are pulled along the expected line of standard.

Assignment Submissions

Assignments are given for new functions/topics taught, resulting in maximum such assignments are to be submitted during the course period.

Preparing the final course report in the prescribed format

During the last fifteen days in the fourth month, students many be asked to consolidate and compile all the assignments submitted in a word document along with the other chapters which will constitute a course report for each student. This report will be the unique contribution of a student carries from the trust to show case the rigorous training he/she received during the period of four months. The report will stand as a testimony for the detailed learning skills of a student which has been acquired in the chosen area. This will facilitate the industry in handpicking the required student for the job.

Attend the whole period of class

All the students are expected to attend the whole period of each and every class. In case if any student is observed moving out of classes after logging in which does not go well with the learning objective of students.

Ensure discipline in the group/class

All the students are advised strictly to follow group/class etiquette and restrain from posting in the group/class any immoral messages or teasing messages or personal interactive messages. This group/class is purely created for academic purpose and hence only academic interactions should go on.

PYTHON AND MACHINE LEARNING BASICS

This course is taught to the students as per the syllabus prescribed and as per the course modalities keeping in mind the guidelines set for them. Periodical tests are conducted to assess the understanding of the students in the course. Assignments are given to ensure that students gain versality in the Data analysis, Data visualization etc. The assignments of the student are given below. These assignments which are done with learning skill and out of box thinking not only reflect the student's innovativeness but also constitute solved exercises in the

Various Data set for the fresh learners to practice and gain confidence. The assignments are listed below in the order of their conducting during the course.

```
sales=3000
profit=2500
```

problem-you have to print the type of sales and profit the type of profit and you have to seggregate those output

In [2]:

```
sales=3000
profit=2500
print('sales',sales)
print('profit',profit)
print('\n')
print(type(sales))
print(type(profit))
```

```
sales 3000
profit 2500


<class 'int'>
<class 'int'>
```

Calculate the simple interest

In [7]:

```
p=int(input('enter a principle-'))
t=int(input('enter a time-'))
r=float(input('enter a rate-'))
si=((p*t*r)/100)
print('\n')
print(si)
```

```
enter a principle-10000
enter a time-5
enter a rate-5.5


2750.0
```

Calculate the area of the triangle

In [8]:

```
base = float(input("Enter the base:"))
height = float(input("Enter the height:"))
area = 0.5*base*height
area
```

```
Enter the base:5
Enter the height:10
Out[8]:
25.0
```

output-p:y:t:h:o:n:

In [9]:

```
string='python'
for i in string:
    print(i,end=':')
```

```
p:y:t:h:o:n:
```

STEP - 1 Score = int(input('Enter your Score - '))

STEP - 2 IF a Student Score is less than or equal to 100 and Greater than or equal to 90(print Distinction)

STEP - 3 Other scores are Distributed Accordingly by Using Nested if-else

HINT : USE RELATIONAL AND LOGICAL OPERATORS.

In [10]:

```
score = int(input('enter your score-'))
if  (score<=100 and score>=90):
    print('distinction')
elif (score<=90 and score>=80):
    print('b')
elif (score<=80 and score>=70):
    print('C')
elif (score<=70 and score>=60):
    print('d')
elif (score<=60 and score>=50):
    print('e')
else:
    print('fail')
```

```
enter your score-70
C
```

print the Message : Calculator

1.Add

2.Subract

3.Multiply

4.Divide

Input : Enter Choice (1-4) : 3

Enter A : 10

Enter B : 20

Output : product = 200

In [11]:

```
print('calculator')
print('1.Add 2.Subract 3.Multiply 4.Divide')
a = int(input('enter your number'))
b = int(input('enter your second number'))
choice = int(input('enter your choice'))
if choice==1:
    print('addition- a+b')
    add=a+b
    print(add)
elif choice==2:
    print('subtraction-a-b')
    subtract=a-b
    print(subtract)

elif choice==3:
    print('multiplication-a*b')
    multiply=a*b
    print(multiply)

elif choice==4:
    print('division-a /b')
    division=a/b
    print(division)

else:
    print('there is no choice')
```

```
calculator
1.Add 2.Subract 3.Multiply 4.Divide
```

```
enter your number2
enter your secondnumber4
enter your choice1
addition-a+b
6
```

```
In [12]:
string1=input('enter your string1')
string2=input('enter your string2')
if string1=='xyz' and string2=='xyz123':
    print('your login is successfull')
else:
    print('password is wrong')
```

```
enter your string1sai
enter your string2lakshmi
password is wrong
```

`Accept the Age, Sex ('M','F'),

numbers of Days and Display the Wages Accordingly. AGE SEX WAGE/DAY

=18 and <30 M,F 700,750

=30 and <=40 M 800

if age does not fall in the range then print error message-'enter appropriate age'`

```
In [ ]:
age=int(input('enter the age'))
sex=input('enter the sex')
print('enter the number of days')
days=int(input('enter the days'))
if (age>=18 and age<30) and (sex.upper()=='M'):
    wages=days*700
    print(wages)
elif (age>=18 and age<30) and (sex.upper()=='F'):
    wages=days*750
    print(wages)
elif (age>=30 and age<=40) and (sex.upper()=='M'):
    wages=days*800
    print(wages)
elif (age>=30 and age<=40) and (sex.upper()=='F'):
    wages=days*850
    print(wages)
```

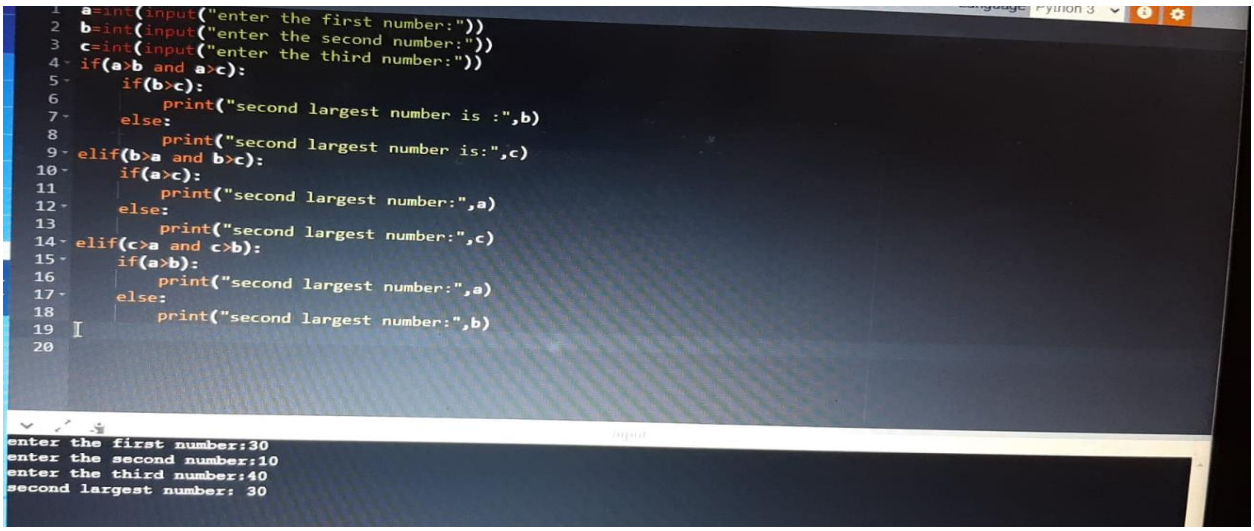
else:

('enter appropriate age')

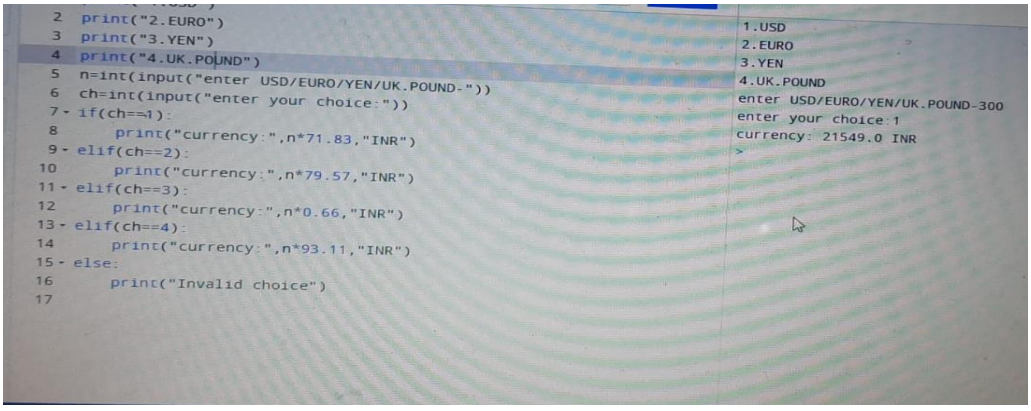
enter the age40
enter the sexF
enter the number of days
enter the days1
850

second largest number among three numbers using both and and or opertators

In [ ]:



Write a program to make a currency converter .The currency converter should be able to convert a specific currency to an appropriate INR value



Accept the following from the user and calculate the percentage of classes attended

1)no of working days

2)no of days absented

3)if the person has less than 75% attendance then the person will not be permitted to the class

```
main.py                                                    Run
1  total_days = int(input("enter the no of working days-"))
2  absent_days = int(input("enter the no of absent days-"))
3  present_days = total_days-absent_days
4  print("present days are:",present_days)
5  percentage = (present_days/total_days)*100
6  print("percentage is:",percentage)
7  if(percentage<75):
8      print("The Student will not be permitted for the Exams")
9  else:
10     print("The Student will  be permitted for the Exams")
11
```

```
Shell
enter the no of working days-10
enter the no of absent days-7
present days are: 3
percentage is: 30.0
The Student will not be permitted for the Exams
>
```

Write a program to find the given number is zero, positive or negative using nested if else

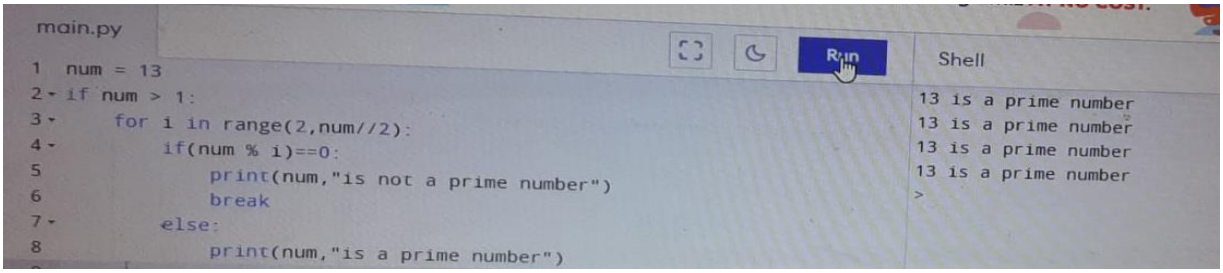
```
print(" number is postive")
else:
    print("number is negative")
```

Check whether the triangle is equilateral, Isosceles or scalene.

```
main.py                                                    Run
1  n1=int(input("enetr the first side:"))
2  n2=int(input("enter the second side:"))
3  n3=int(input("enterthe third side:"))
4  if(n1==n2)and(n2==n3)and(n3==n1):
5      print("The given triangle is equilateral")
6  elif((n1==n2)and(n2!=n3))or((n2==n3)and(n2!=n1))or((n3==n1)and(n3!=n2
       )):
7      print("the given triangle is isoceles")
8  elif(n1!=n2)and(n2!=n3)and(n3!=n1):
9      print("the Triangle is scalene")
10 else:
11     print("invalid")
```

```
Shell
enetr the first side:1
enter the second side:2
enterthe third side:1
the given triangle is isoceles
>
```

Write a program to find whether the given number is prime or not.



Write a program to get the following

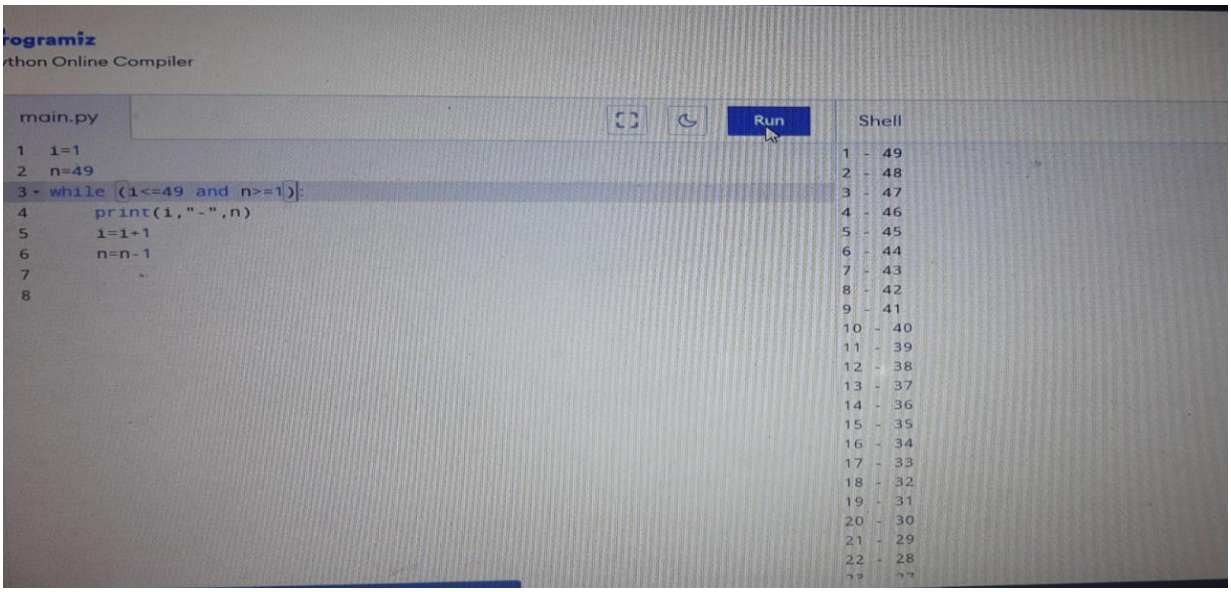
1-49

2-48

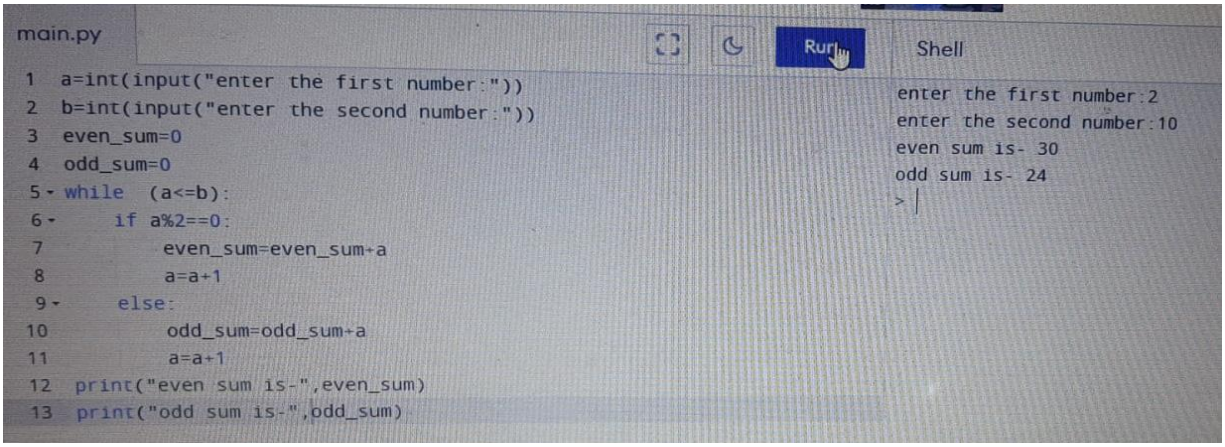
…….

48-2

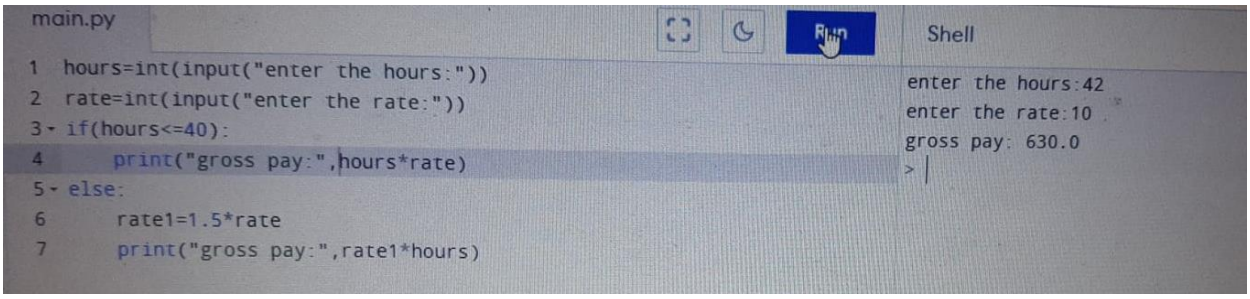
49-1



Write a program to display sum of even and sum of odd numbers.



Write a program to accept hours and rate per hour from the user and compute the gross pay based on the following.



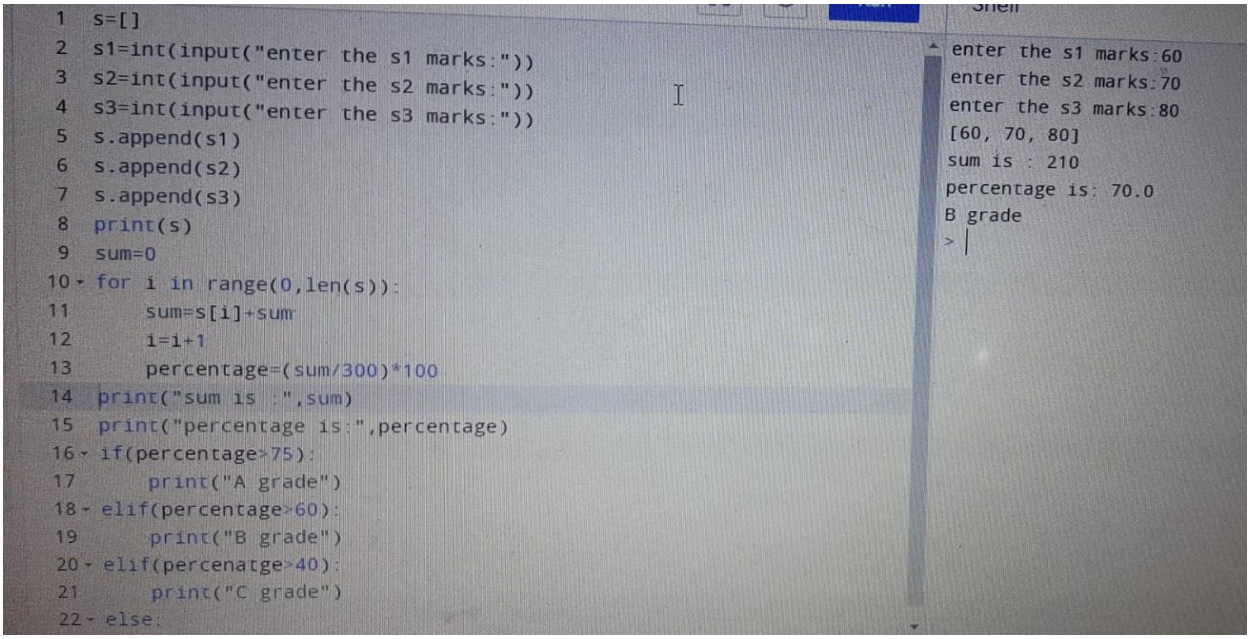
Write a program to print grade of the student.

Take marks of a student in 3 subjects as input and store them in a list

Calculate the percentage out of 300:

Print grade based on the following condition

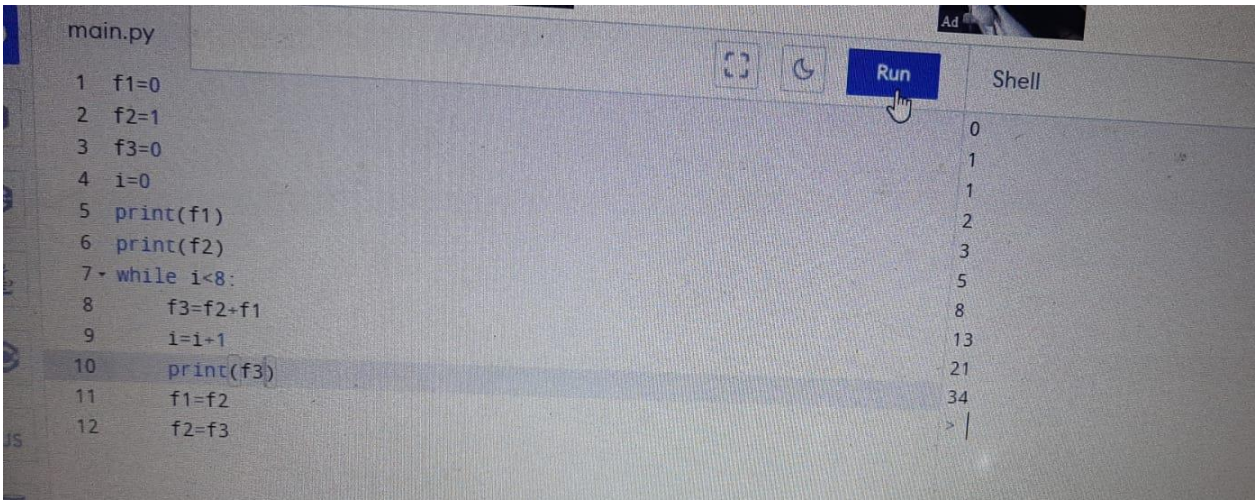
>75-A grade >60-B grade 40-C grade



```
s=[]
s1=int(input("enter the s1 marks:"))
s2=int(input("enter the s2 marks:"))
s3=int(input("enter the s3 marks:"))
s.append(s1)
s.append(s2)
s.append(s3)
print(s)
sum=0
for i in range(0,len(s)):
    sum=s[i]+sum
    i=i+1
    percentage=(sum/300)*100
print("sum is :",sum)
print("percentage is:",percentage)
if(percentage>75):
    print("A grade")
elif(percentage>60):
    print("B grade")
elif(percenatge>40):
    print("C grade")
else:
```

```
enter the s1 marks:60
enter the s2 marks:70
enter the s3 marks:80
[60, 70, 80]
sum is : 210
percentage is: 70.0
B grade
>
```

Write a program for generating the F-Series



main.py

```
f1=0
f2=1
f3=0
i=0
print(f1)
print(f2)
while i<8:
    f3=f2+f1
    i=i+1
    print(f3)
    f1=f2
    f2=f3
```

Run

Shell

```
0
1
1
2
3
5
8
13
21
34
>
```

Create a dictionary and access it values using a condition on its key. The data is given below. let the condition on the key is that it should be a even number

```
dict={1:"aman",
    2:"mohith",
```

3:"gauri",

4:"imran",

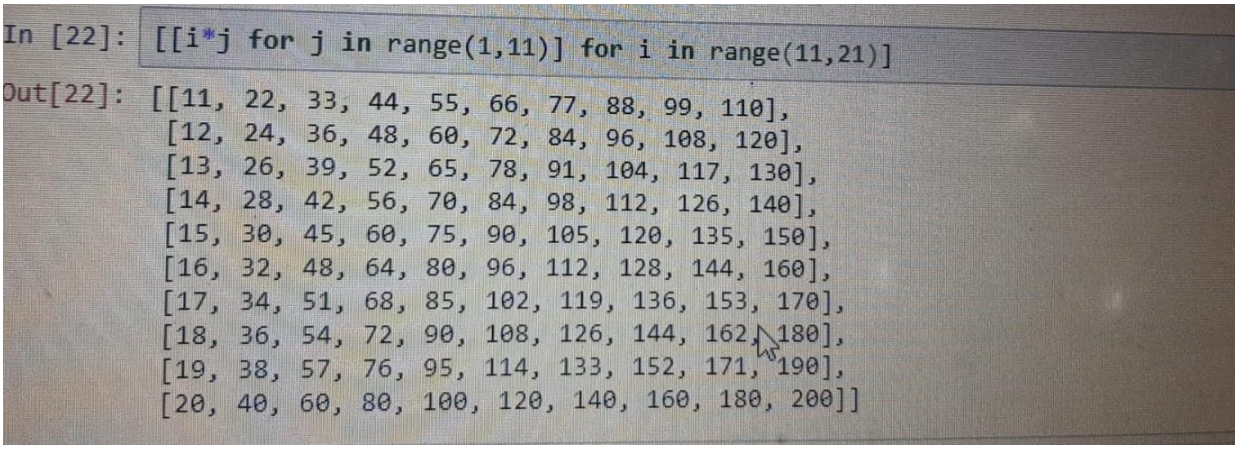
5:"roma"

}

{i:dict[i] for i in dict if i%2==0}

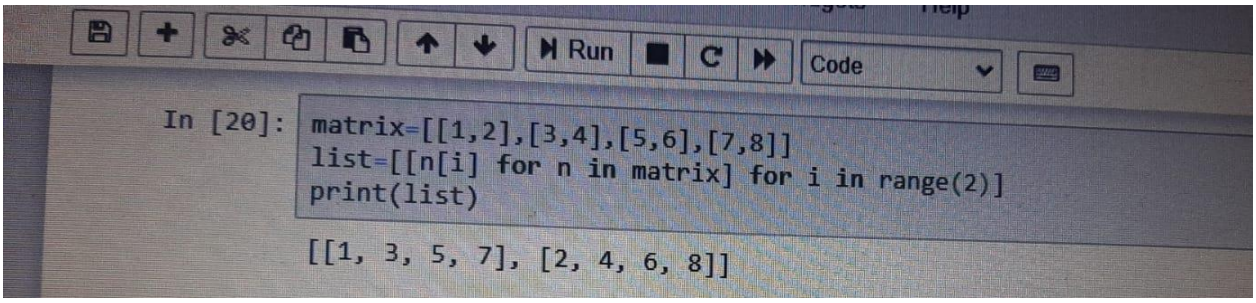
```
{2: 'mohith', 4: 'imran'}
```

Use nested list comprehension to print table of 11 to 20.



```
In [22]: [[i*j for j in range(1,11)] for i in range(11,21)]

Out[22]: [[11, 22, 33, 44, 55, 66, 77, 88, 99, 110],
          [12, 24, 36, 48, 60, 72, 84, 96, 108, 120],
          [13, 26, 39, 52, 65, 78, 91, 104, 117, 130],
          [14, 28, 42, 56, 70, 84, 98, 112, 126, 140],
          [15, 30, 45, 60, 75, 90, 105, 120, 135, 150],
          [16, 32, 48, 64, 80, 96, 112, 128, 144, 160],
          [17, 34, 51, 68, 85, 102, 119, 136, 153, 170],
          [18, 36, 54, 72, 90, 108, 126, 144, 162, 180],
          [19, 38, 57, 76, 95, 114, 133, 152, 171, 190],
          [20, 40, 60, 80, 100, 120, 140, 160, 180, 200]]
```

Use list comprehension to find transpose of the matrix.



```
Run   Code

In [20]: matrix=[[1,2],[3,4],[5,6],[7,8]]
         list=[[n[i] for n in matrix] for i in range(2)]
         print(list)

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

The days should be keys and the the temperature in Celsius corresponding to the days should be values. Hence create a dictionary comprehension by using list comprehension.

```
[6]: days=["sunday","monday","tuesday","wednesday","thursday","friday","saturday"]
     temp_c=[30.5,32.1,30.2,29.9,30.9,29.7,31.8]
     my_dict={i:j for(i,j) in zip(days,temp_c)}
     my_dict

t[6]: {'sunday': 30.5,
       'monday': 32.1,
       'tuesday': 30.2,
       'wednesday': 29.9,
       'thursday': 30.9,
       'friday': 29.7,
       'saturday': 31.8}
```

Customers:Alex,Bob,Carol,Dave,Flow,katie,Nate

Part1- Lets say we have a list of customers who visit our store, and we would like to offer a random discount to each customer. We would like the discount value to be anywhere between $1 and $100.

HINT: JUST IMPORT RANDOM LIBRARY TO FETCH VALUES

Part2- We would know like to offer customers who were offered fewer than 30% discount a 10% discount on the next purchase will be added.

```
[21]: import random
      customers=["alex","Bob Carol","Dave","Flow","Katie","nate"]
      discount={customer:random.randint(1,100) for customer in customers}
      print(discount)
      customer={customers:discount for(customers,discount) in discount.items()if discount<30}
      print(customer)

      {'alex': 12, 'Bob Carol': 5, 'Dave': 13, 'Flow': 52, 'Katie': 41, 'nate': 2}
      {'alex': 12, 'Bob Carol': 5, 'Dave': 13, 'nate': 2}
```

Write a multiplication program to accept range of numbers?

```
def multiplication(*range):
    product=1
    for var in range:
        product=product*var
    print(product)
multiplication(8,2,3,-1,7)
```

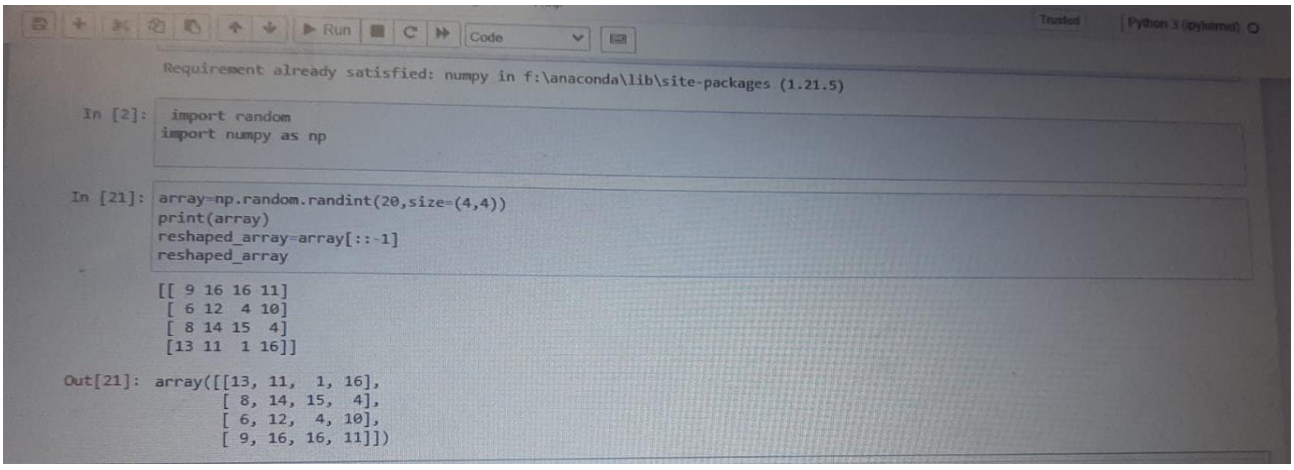
-336

Write a program to check whether the number is prime or not using functions

```
def prime(n):
  flag=1
if n>1:
    for i in range(2,n):
            if n%i==0:
                flag==0
                break
            else:
                flag==1
else:
   print('enter a valid number')
if flag==0:
  print(' not prime')
else:
   print(' prime')
prime(5)
```

```
prime
```

Write a program to create a 4*4 array with random values, now create a new array from the said array such that it starts with the last row and goes till the end.



Write a program to calculate the factorial of a number

```
In [10]: def fact(n):
             if n==1:
                 return 1
             else:
                 return (n*fact(n-1))
         n=int(input("enter the number:"))
         print("factorial of",n,"is",fact(n))

         enter the number:5
         factorial of 5 is 120
```

Write a define python function to find the max of three numbers.

```
def max(a,b,c):
    if a>b and a>c:
        print(a,"is the greatest number")
    elif b>a and b>c:
        print(b,"is the greatest number")
    else:
        print(c,"is the greatest number")
a=int(input("enter the first number:"))
b=int(input("enter the second number:"))
c=int(input("enter the third number:"))
max(a,b,c)

enter the first number:2
enter the second number:8
enter the third number:0
8 is the greatest number
```

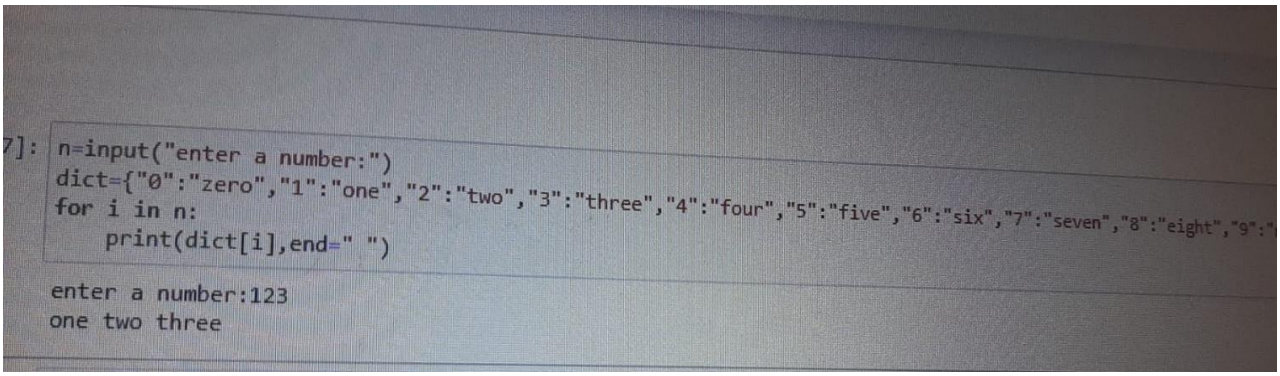
Write a program to define function to create and print a list where the values are squared of numbers Between 1 and 30(both included)

```
In [30]: def num():
             s=[]
             for i in range(1,31):
                 s.append(i*i)
             print(s)

In [31]: num()
         [1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 78
         4, 841, 900]
```

Reverse the list by using define function with proper coding.

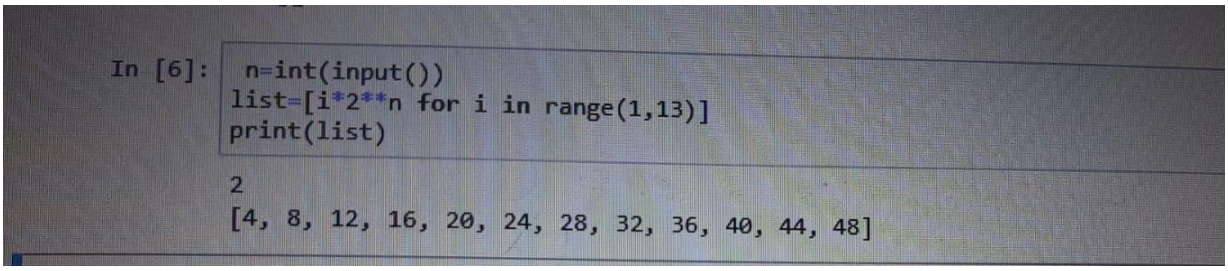
```
In [34]: def rev():
             s=[2,4,6,3,5,2,6,43]
             a=s[::-1]
             print(a)

In [35]: rev()
         [43, 6, 2, 5, 3, 6, 4, 2]
```

Write a program to convert number into alphabetic format?



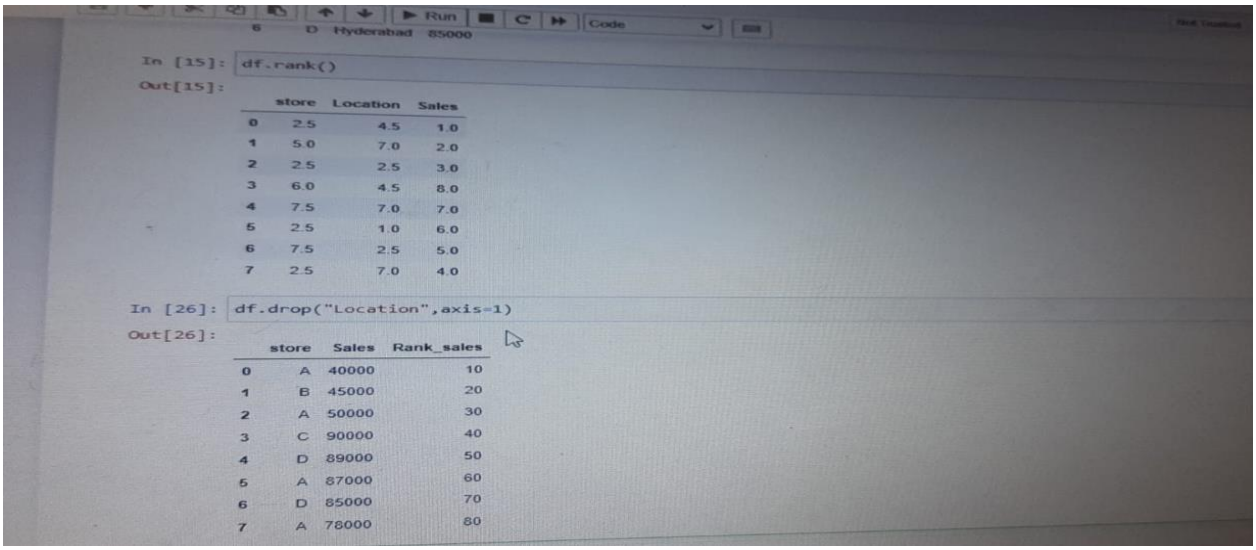
```
7]: n=input("enter a number:")
dict={"0":"zero","1":"one","2":"two","3":"three","4":"four","5":"five","6":"six","7":"seven","8":"eight","9":"
for i in n:
    print(dict[i],end=" ")

enter a number:123
one two three
```

Write a program to print the entered number table?



```
In [6]: n=int(input())
list=[i*2**n for i in range(1,13)]
print(list)

2
[4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48]
```

Drop the location column form the dataset?



```
6  D  Hyderabad  85000

In [15]: df.rank()
Out[15]:
```

	store	Location	Sales
0	2.5	4.5	1.0
1	5.0	7.0	2.0
2	2.5	2.5	3.0
3	6.0	4.5	8.0
4	7.5	7.0	7.0
5	2.5	1.0	6.0
6	7.5	2.5	5.0
7	2.5	7.0	4.0

```
In [26]: df.drop("Location",axis=1)
Out[26]:
```

	store	Sales	Rank_sales
0	A	40000	10
1	B	45000	20
2	A	50000	30
3	C	90000	40
4	D	89000	50
5	A	87000	60
6	D	85000	70
7	A	78000	80

Write a program to print the number of candies present in the problem?

```
[7]: n=int(input("enter the number of candies:"))
     for i in range(n):
         if i%5==2:
             if i%6==3:
                 if i%7==2:
                     print("no of candies in the bowl are:",i)

enter the number of candies:200
no of candies in the bowl are: 177
```

Plotting:

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
df = pd.read_csv('iris.csv')
df
```

Out[2]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
..	..	..	..	..	..
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

149 rows × 5 columns

In [3]:

```
df.head()
```

Out[3]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

In [4]:

```
df.tail()
```

Out[4]:

	5.1	3.5	1.4	0.2	Iris-setosa
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

In [7]:

```
df.shape
```

Out[7]:

```
(149, 5)
```

In [9]:

```
df.dtypes
```

Out[9]:

```
5.1            float64
3.5            float64
1.4            float64
0.2            float64
Iris-setosa     object
dtype: object
```

The number of insects in a lab doubles in size every month. Take the initial number of insects as input and output a list, showing the number of insects for each of the next 12 months, starting with 0, which is the initial value. So, the resulting list should contain 12 items, each showing the number of insects at the beginning of that month. (BY USING LIST COMPREHENSION)

In [12]:

```
num=int(input("enter:"))
li = [i*2**num for i in range(1,12)]
print(li)
```

```
enter:3
[8, 16, 24, 32, 40, 48, 56, 64, 72, 80, 88]
```

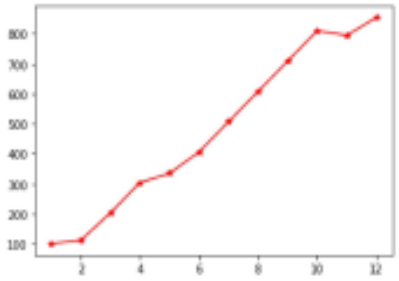
line plots:

In [1]:

```
#visualization
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

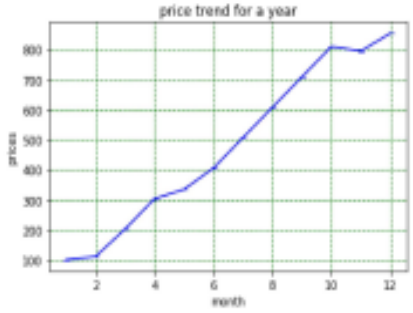
In [6]:

```
#create a data
month=np.arange(1,13)
prices=[101,112,203,304,335,406,507,608,709,810,795,854]
plt.plot(month,prices,color='r',marker='*')
#displying the data
plt.show()
```

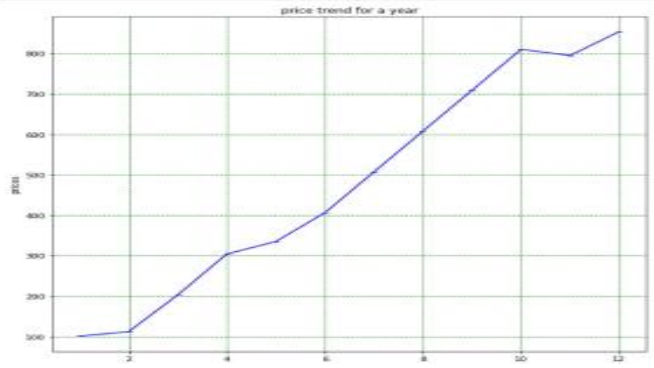


In [21]:

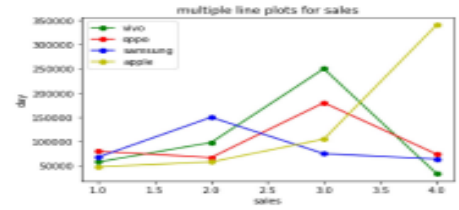
```
month=np.arange(1,13)
prices=[101,112,203,304,335,406,507,608,709,810,795,854]
plt.plot(month,prices,color='b',marker='_')
#title
plt.title('price trend for a year')
#axis labels
plt.xlabel("month")
plt.ylabel("prices")
#grid lines
plt.grid()
#grid color
plt.grid(color='g',ls='--')
plt.show()
```



```
#figure size:
plt.figure(figsize=(10,10))
month=np.arange(1,13)
prices=[101,112,203,304,335,405,507,608,709,810,795,854]
plt.plot(month,prices,color='b',marker='_')
#title
plt.title('price trend for a year')
#axis labels
plt.xlabel("month")
plt.ylabel("prices")
#grid lines
plt.grid()
#grid color
plt.grid(color='g',ls='--')
plt.show()
```




```
#multiple line plots:
day=[1,2,3,4]

vivo_sales=[58000,98000,250000,34000]
oppo_sales=[80000,67000,180000,74000]
samsung_sales=[68000,150000,75000,60000]
apple_sales=[40000,58000,105000,140000]

plt.plot(day,vivo_sales,color='g',label='vivo',marker='o')
plt.plot(day,oppo_sales,color='r',label='oppo',marker='o')
plt.plot(day,samsung_sales,color='b',label='samsung',marker='o')
plt.plot(day,apple_sales,color='y',label='apple',marker='o')

#add axes and plot labels
plt.title('multiple line plots for sales')
plt.xlabel("sales")
plt.ylabel("day")
#add the legend
plt.legend()
plt.show()
```



scatter plot

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=sns.load_dataset('iris')
df
```

Out[1]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [2]:

```
df.head()
```

Out[2]:

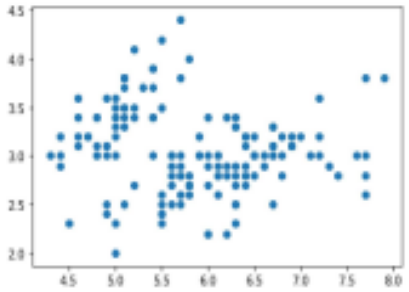
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [3]:

```
plt.scatter(x='sepal_length',y='sepal_width',data=df)
```

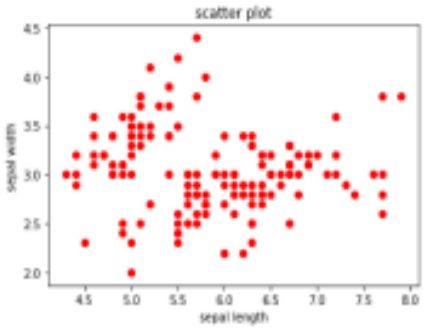
Out[3]:

```
<matplotlib.collections.PathCollection at 0x253f3100648>
```



In [6]:

```
plt.scatter(x='sepal_length',y='sepal_width',data=df,color='r')
#add axes
plt.title("scatter plot")
plt.xlabel("sepal length")
plt.ylabel("sepal width")

#display the plot
plt.show()
```



multiple scatter plot

In [7]:

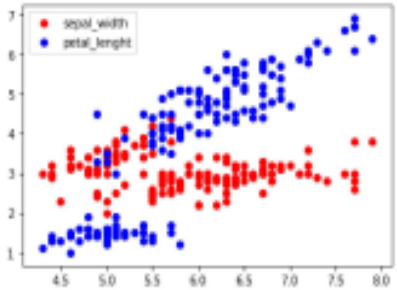
```
df.head(1)
```

Out[7]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa

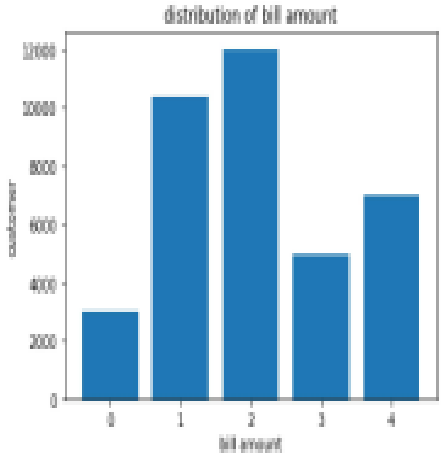
In [9]:

```
plt.scatter(x='sepal_length',y='sepal_width',label='sepal_width',color='r',data=df)
plt.scatter(x='sepal_length',y='petal_length',label='petal_lenght',color='b',data=df)
plt.legend()
plt.show()
```



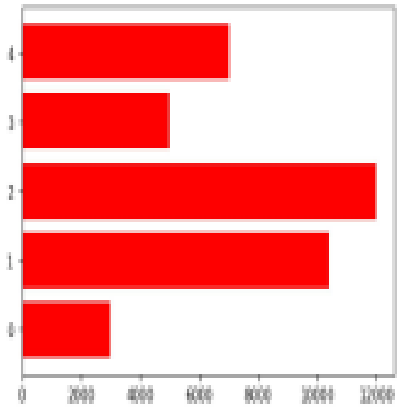
In [10]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

amount=[3000,10340,12000,5000,7000]
customer=('roja','sunil','sai','charitha','sneha')
x_pos=np.arange(len(customer))

plt.bar(x = x_pos, height=amount)
#add names
plt.title("distribution of bill amount")
plt.xlabel("bill amount")
plt.ylabel("customer")
plt.show()
```



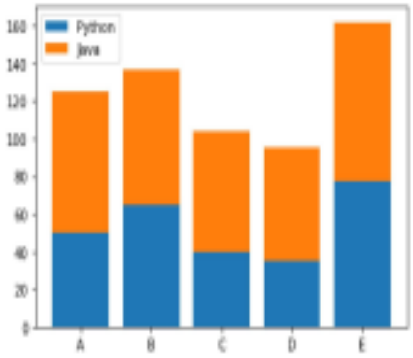
In [8]:

```
y_pos=np.arange(len(customer))

plt.barh(y = y_pos, width=amount,color='r')
plt.show()
```



In [9]:

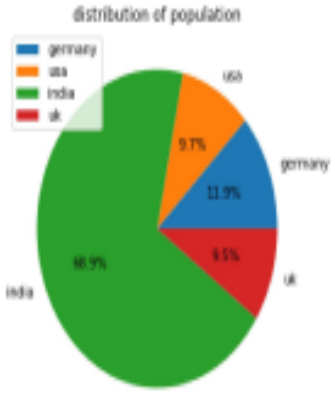
```
# Create a Data:
python_marks = (50, 65, 40, 35, 77)
java_marks = (75, 72, 64, 60, 85)

# Set the Position of the Bar:
index = np.arange(5)

plt.bar(x = index, height = python_marks, label = 'Python')
plt.bar(x = index, height = java_marks, bottom = python_marks, label = 'Java')

plt.xticks(ticks = index, labels = ('A', 'B', 'C', 'D', 'E'))

plt.legend()

plt.show()
```



pie plot

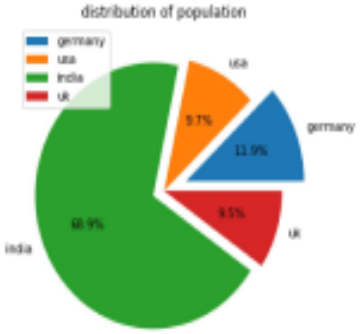
In [13]:

```
plt.figure(figsize=(5,5))
countries={'germany','usa','india','uk'}
population=[8.26,6.7,47.8,6.6]
plt.pie(x=population,labels=countries,autopct='%1.1f%%')
plt.title('distribution of population')
plt.legend()
plt.show()
```



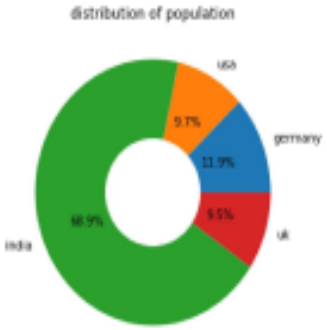
In [7]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(5,5))

countries=['germany','usa','india','uk']
population=[8.26,6.7,47.8,6.6]

#if dont give autopct ntg will displys
plt.pie(x=population,labels=countries,autopct='%1.1f%%' ,explode = explode)
#adding titles:
plt.title('distribution of population')
plt.legend()

#explode pie plot:
explode=(0.2,0,0.1,0)
plt.show()
```



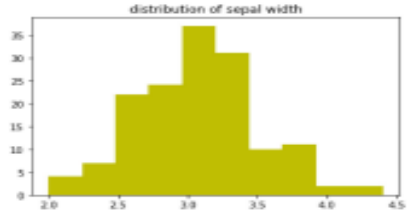
In [21]:

```
plt.figure(figsize=(5,5))

countries=['germany','usa','india','uk']
population=[8.26,6.7,47.8,6.6]

#if dont give autopct ntg will displys
plt.pie(x=population,labels=countries,autopct='%1.1f%%')

#donutpie
circle=plt.Circle(xy = (0, 0),radius=0.4,color='w')
plt.gcf()
plt.gca().add_artist(circle)

#adding titles:
plt.title('distribution of population')

plt.show()
```



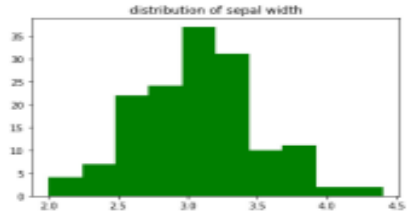
In [25]:

```
df=sns.load_dataset('iris')
df.head()

plt.hist(x=df.sepal_width,color='y')
plt.title('distribution of sepal width')
plt.show()
```



In [28]:

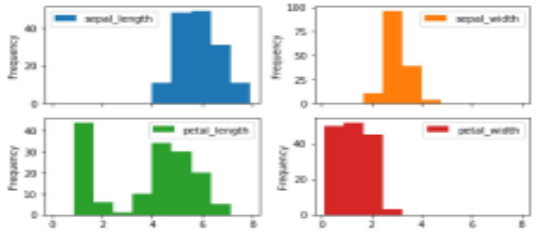
```
plt.hist(x=df.sepal_width,color='g',bins=10)
plt.title('distribution of sepal width')
plt.show()
```



histogram

In [4]:

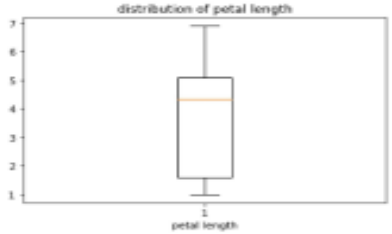
```
#multiple histogram
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df=sns.load_dataset('iris')
df.head()

df.plot.hist(subplots=True,layout=(2,2),figsize=(7,4),
             sharex=True,sharey=False)
plt.tight_layout()
plt.show()
```



boxplot

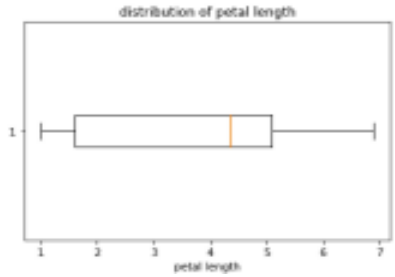
In [5]:

```
plt.boxplot(x=df.petal_length)
plt.title('distribution of petal length')
plt.xlabel('petal length')
plt.show()
```



In [6]:

```
plt.boxplot(x=df.petal_length,vert=False)
plt.title('distribution of petal length')
plt.xlabel('petal length')
plt.show()
```




```
[ ]: #ASSIGNMENT:
     #Input:program to count the number of vowels in the string
     #output:

[5]: n=input("enter a string:")
     c=0
     for i in n:
         if (i=='a' or i=='e' or i=='i' or i=='o' or i=='u'):
             c=c+1

     print("no of vowels in the string is:",c)

     enter a string:sai
     no of vowels in the string is: 2
```

Run | Code

```
In [ ]: #assignment
        #You are making a text editor and need to implement·find/replace functionality. The co
        #For example, for the given text "I weigh 80 kilograms. He weighs 65 kilograms.":

n [15]: text = "I weighs 80 kilograms"
        string1 = input("enter the string1:")
        string2 = input("enter the string2:")
        print(text.replace(string1, string2))
        print("no of replacements made:",text.count(string1))

        enter the string1:I
        enter the string2:He
        He weighs 80 kilograms
        no of replacements made: 1
```

Write a program to remove the first occurrence of an element an d find the missing element form an array?

```
In [72]: #Write a Python program to remove the first occurrence of a specified element from an arr
         a = array('i', [1, 3, 5, 3, 7, 1, 9, 3])
         print("original array:",a)
         for i in a:
         a.remove(3)
         print("New array:",a)

         original array: array('i', [1, 3, 5, 3, 7, 1, 9, 3])
         New array: array('i', [1, 5, 3, 7, 1, 9, 3])
```

```
In [11]: a= np.array([10,11,12,13,15,16,17,18,19,20])
         print("original array:",a)
         for i in range(10,21):
             if i not in a:
                 print("missing number:",i)

         original array: [10 11 12 13 15 16 17 18 19 20]
         missing number: 14
```

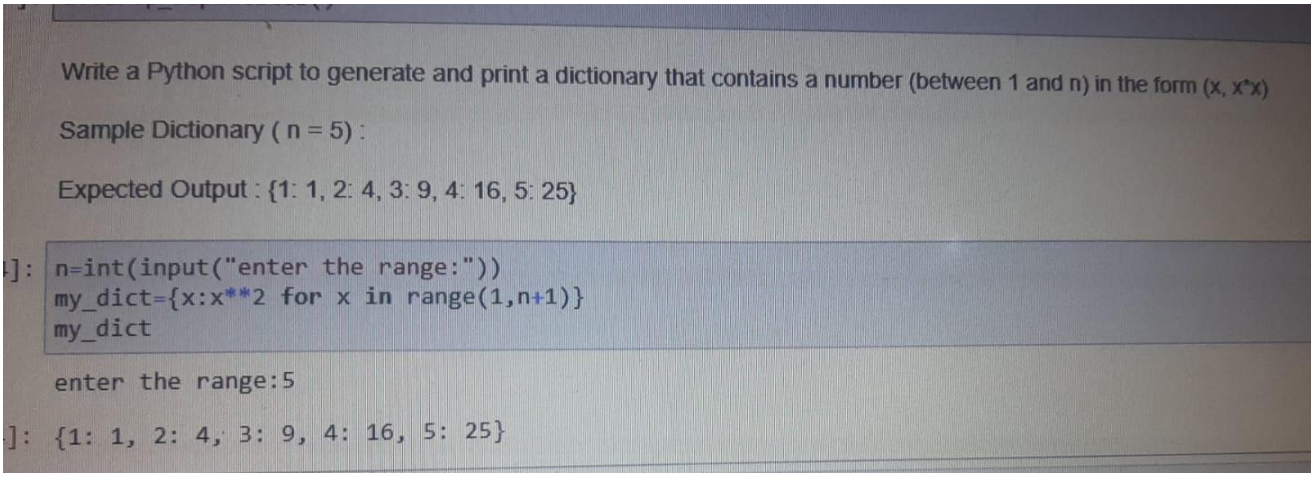
Define a function that accepts two strings as input and print the string with maximum length?

```
In [17]: #Define a Function that can Accept two strings as input and print the String with maximum lengt
         def maxstring(s1,s2):
             a=len(s1)
             b=len(s2)
             if a>b:
                 print(s1)
             elif a<b:
                 print(s2)
             else:
                 print(s1)
                 print(s2)
         s1=input("enter teh string1:")
         s2=input("enter teh string2:")
         maxstring(s1,s2)

         enter teh string1:sai
         enter teh string2:lakshmi
         lakshmi
```

Convert a string to an integer by using define function?

```
[26]: #Convert a String to an Integer by using Define Function.
      def integer(a,b):
          print(int(a),int(b))

[27]: integer('45','51')

      45 51
```

Write a python script and print a dictionary that contains a number (between 1 and n) in the form(x,x*x)?



statistical summary on individual data set:

In [3]:
```
df_placements=pd.read_csv('Placement_Data_Full_Class.csv')
df_placements
```

Out[3]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210	211	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Placed	400000.0
211	212	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	74.0	Mkt&Fin	53.62	Placed	275000.0
212	213	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	59.0	Mkt&Fin	69.72	Placed	295000.0
213	214	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	70.0	Mkt&HR	60.23	Placed	204000.0
214	215	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	89.0	Mkt&HR	60.22	Not Placed	NaN

215 rows × 15 columns

In [4]:
```
df_placements.head()
```

Out[4]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0

In [5]:
```
df_placements.tail()
```

Out[5]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
210	211	M	80.6	Others	82.0	Others	Commerce	77.6	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Placed	400000.0
211	212	M	58.0	Others	60.0	Others	Science	72.0	Sci&Tech	No	74.0	Mkt&Fin	53.62	Placed	275000.0
212	213	M	67.0	Others	67.0	Others	Commerce	73.0	Comm&Mgmt	Yes	59.0	Mkt&Fin	69.72	Placed	295000.0
213	214	F	74.0	Others	66.0	Others	Commerce	58.0	Comm&Mgmt	No	70.0	Mkt&HR	60.23	Placed	204000.0
214	215	M	62.0	Central	58.0	Others	Science	53.0	Comm&Mgmt	No	89.0	Mkt&HR	60.22	Not Placed	NaN

In [6]:
```
df_placements.describe()
```

Out[6]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	148.000000
mean	108.000000	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
std	62.209324	10.827205	10.897509	7.358743	13.275956	5.833385	93457.452420
min	1.000000	40.890000	37.000000	50.000000	50.000000	51.210000	200000.000000
25%	54.500000	60.600000	60.900000	61.000000	60.000000	57.945000	240000.000000
50%	108.000000	67.000000	65.000000	66.000000	71.000000	62.000000	265000.000000
75%	161.500000	75.700000	73.000000	72.000000	83.500000	66.255000	300000.000000
max	215.000000	89.400000	97.700000	91.000000	98.000000	77.890000	940000.000000

In [7]:
```
df_placements.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215 entries, 0 to 214
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   sl_no           215 non-null    int64  
 1   gender          215 non-null    object 
 2   ssc_p           215 non-null    float64
 3   ssc_b           215 non-null    object 
 4   hsc_p           215 non-null    float64
 5   hsc_b           215 non-null    object 
 6   hsc_s           215 non-null    object 
 7   degree_p        215 non-null    float64
 8   degree_t        215 non-null    object 
 9   workex          215 non-null    object 
 10  etest_p         215 non-null    float64
 11  specialisation  215 non-null    object 
 12  mba_p           215 non-null    float64
 13  status          215 non-null    object 
 14  salary          148 non-null    float64
dtypes: float64(6), int64(1), object(8)
memory usage: 25.3+ KB
```

In [8]:
```
df_placements.duplicated()
```

Out[8]:

```
0      False
1      False
2      False
3      False
4      False
       ...
210    False
211    False
212    False
213    False
214    False
Length: 215, dtype: bool
```

In [9]:
```
df_placements.duplicated().sum()
```

Out[9]:

```
0
```

In [10]:
```
df_placements.shape
```

Out[10]:
```
(215, 15)
```

In [11]:
```
df_placements.dtypes
```

Out[11]:
```
sl_no              int64
gender            object
ssc_p            float64
ssc_b             object
hsc_p            float64
hsc_b             object
hsc_s             object
degree_p         float64
degree_t          object
workex            object
etest_p          float64
specialisation    object
mba_p            float64
status            object
salary           float64
dtype: object
```

In [12]:
```
df_placements.isnull()
```

Out[12]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	True
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
211	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
212	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
213	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
214	False	False	False	False	False	False	False	False	False	False	False	False	False	False	True

In [13]:
```
df_placements.isnull().sum()
```

Out[13]:
```
sl_no             0
gender            0
ssc_p             0
ssc_b             0
hsc_p             0
hsc_b             0
hsc_s             0
degree_p          0
degree_t          0
workex            0
etest_p           0
specialisation    0
mba_p             0
status            0
salary           67
dtype: int64
```

In [14]:
```
df_placements.isnull().sum().sum()
```

Out[14]:
```
67
```

In [43]:
```
df_placements['gender'].value_counts()
```

Out[43]:
```
M    139
F     76
Name: gender, dtype: int64
```

In [44]:
```
df_placements['hsc_s'].value_counts()
```

Out[44]:
```
Commerce    113
Science      91
Arts         11
Name: hsc_s, dtype: int64
```

In [45]:
```
#create dummy variables for gender:
#drop_first = 'True' creates(n-1) dummy variables from categories

pd.get_dummies(df_placements,columns=['gender'], drop_first = True)
```

Out[45]:

	sl_no	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary	gender_M
0	1	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0	1
1	2	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0	1
2	3	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0	1
3	4	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN	1
4	5	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210	211	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Placed	400000.0	1
211	212	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	74.0	Mkt&Fin	53.62	Placed	275000.0	1
212	213	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	59.0	Mkt&Fin	69.72	Placed	295000.0	1
213	214	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	70.0	Mkt&HR	60.23	Placed	204000.0	0
214	215	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	89.0	Mkt&HR	60.22	Not Placed	NaN	1

215 rows × 15 columns

In [46]:
```
pd.get_dummies(df_placements,columns=['hsc_s'], drop_first = True)
```

Out[46]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary	hsc_s_Commerce	hsc_s_Science
0	1	M	67.00	Others	91.00	Others	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0	1	0
1	2	M	79.33	Central	78.33	Others	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0	0	1
2	3	M	65.00	Central	68.00	Central	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0	0	0
3	4	M	56.00	Central	52.00	Central	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN	0	1
4	5	M	85.80	Central	73.60	Central	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210	211	M	80.60	Others	82.00	Others	77.60	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Placed	400000.0	1	0
211	212	M	58.00	Others	60.00	Others	72.00	Sci&Tech	No	74.0	Mkt&Fin	53.62	Placed	275000.0	0	1

ENCODING

In [3]:
```
#one hot encoding:

#import OneHotEncoder from sklearn:
from sklearn.preprocessing import OneHotEncoder

#creating an instance of onehotencoder
ohe = OneHotEncoder()

#fit_transform():fit to data and returns the transformed version
#toarray():returns the numpy array
#columns:add the column names
df_encode = pd.DataFrame(ohe.fit_transform(df[['gender']]).toarray(),
columns = ['gender_M','gender_f'])

#merge with main Dataframe(df_placements)
#axis = 1:it stands for columns
df_encode = pd.concat([df,df_encode], )

#print the first two rows of the dataset
df_encode.head()
```

Out[3]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary	gender_M	gender_f
0	1.0	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0	NaN	NaN
1	2.0	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0	NaN	NaN
2	3.0	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0	NaN	NaN
3	4.0	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN	NaN	NaN
4	5.0	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0	NaN	NaN

`Interpretation` : two new features are added to the dataset based on the unique values in the categorical feature

the two features are 'gender_M' and 'gender_F'

In [4]:
```
#label encoding:

#import LabelEncoder from sklearn:
from sklearn.preprocessing  import LabelEncoder

#create instance:
label_encode = LabelEncoder()

#fit and transform:
df['label_encoded_performance']=label_encode.fit_transform(df.gender)

#display the data
df
```

Out[4]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary	label_encoded_performance
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0	1
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0	1
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0	1
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN	1
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210	211	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Placed	400000.0	1
211	212	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	74.0	Mkt&Fin	53.62	Placed	275000.0	1
212	213	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	59.0	Mkt&Fin	69.72	Placed	295000.0	1
213	214	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	70.0	Mkt&HR	60.23	Placed	204000.0	0
214	215	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	89.0	Mkt&HR	60.22	Not Placed	NaN	1

215 rows × 16 columns

`Interpretation` : as we know that label encoder ranks the categorical variables based on the alphabetical order

hence the categorical variable 'F' is ranked as 0 and 'M' is ranked as 1

In [5]:
```
#ordinal encoding:
#import OrdinalEncoder from the sklearn
from sklearn.preprocessing import OrdinalEncoder

#create an instance
ordinalencode = OrdinalEncoder(categories=[['M','F']])

#fit and transform
df['ordinal_encoded_performance'] = ordinalencode.fit_transform(df.gender.values.reshape(-1,1))

#display the result
df
```

Out[5]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary	label_encoded_performance	ordinal_encoded_performance
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0	1	0.0
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0	1	0.0
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0	1	0.0
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN	1	0.0
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0	1	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210	211	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Placed	400000.0	1	0.0
211	212	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	74.0	Mkt&Fin	53.62	Placed	275000.0	1	0.0
212	213	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	59.0	Mkt&Fin	69.72	Placed	295000.0	1	0.0
213	214	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	70.0	Mkt&HR	60.23	Placed	204000.0	0	1.0
214	215	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	89.0	Mkt&HR	60.22	Not Placed	NaN	1	0.0

215 rows × 17 columns

`Interpretation` : it maps each unique value with an integer value.

therefore 'F' is mapped with 0 and 'M' is mapped with 1

In [6]:
```
#Frequency Encoding:

#size of the each category:
encoding = df.groupby('gender').size()

#get the frequncy of each category:
encoding = encoding / len(df)

#
df['frequency_gender'] = df.gender

#here fit_transform is not done here because reducing columns is needed
# map is used it maps the related datapoints

#display the data
df.head()
```

Out[6]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary	label_encoded_performance	ordinal_encoded_performance	f
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0	1	0.0	
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0	1	0.0	
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0	1	0.0	
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN	1	0.0	
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0	1	0.0	

`Interpretation` : here the frequency of the categories is used as labels

Scaling:

standard scaler

In [14]:
```
#also known as standard scaler or z-score scaler or data normalization
### import standard scaler:
from sklearn.preprocessing import StandardScaler

#minimum and maximum values og=f weight:
#'\n' -add space

print('Minimum value before transformation: ',df.salary.min(),'\n'
      'maximum value before transformation:',df.salary.max(),'\n')

#instantaniate the StandardScaler:
standard_scale = StandardScaler()

#fit the standadrdscaler:
#fit_transform():returns a transformed data
df['Scaled_salary'] = standard_scale.fit_transform(df[['salary']])

print('Minimum value before transformation: ',df['Scaled_salary'].min(),'\n'
      'maximum value before transformation:',df['Scaled_salary'].max(),'\n')
```

```
Minimum value before transformation:  200000.0
maximum value before transformation: 940000.0

Minimum value before transformation:  -0.9518389491566408
maximum value before transformation: 6.993089159343879
```

In [15]:
```
print('Mean :',df['Scaled_salary'].mean())
print('\n')
print('Standard Deviation:',df['Scaled_salary'].std())
```

```
Mean : 2.482998791560316e-16


Standard Deviation: 1.0033955955097846
```

`Interpreation` : it is used to resize the distribution

so that the mean mean is 0 and the standard deviation is 1

In [ ]:
```
##### MinMax Scaler:
```

In [ ]:
```
#import MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

#creating an instance:
min_max = MinMaxScaler()

#fit the min_max scaler
df['min_max_scaled_salary'] = min_max.fit_transform(df[['salary']])

#minimum and maximum of normalized weight:
df['min_max_scaled_salary'].min(),df['min_max_scaled_salary'].max()
```

`Interpretation` :

The new data has been generated for the particular column weight b/w 0-1

Detecting outliers and removing outliers:

In [8]:
```
#also known as standard scaler or z-score scaler or data normalization
### import standard scaler:
from sklearn.preprocessing import StandardScaler

#minimum and maximum values og=f ssc_p:
#'\n' -add space

print('Minimum value before transformation: ',df_placements.ssc_p.min(),'\n'
      'maximum value before transformation:',df_placements.ssc_p.max(),'\n')

#instantaniate the StandardScaler:
standard_scale = StandardScaler()

#fit the standadrdscaler:
#fit_transform():returns a transformed data
df_placements['Scaled_ssc_p'] = standard_scale.fit_transform(df_placements[['ssc_p']])

print('Minimum value before transformation: ',df_placements.ssc_p.min(),'\n'
      'maximum value before transformation:',df_placements.ssc_p.max(),'\n')
```

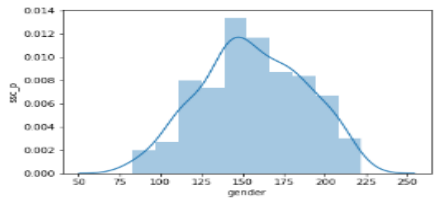
```
Minimum value before transformation:  40.89
maximum value before transformation: 89.4

Minimum value before transformation:  40.89
maximum value before transformation: 89.4
```

In [9]:
```
#import MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

#creating an instance:
min_max = MinMaxScaler()

#fit the min_max scaler
df_placements['min_max_scaled_ssc_p'] = min_max.fit_transform(df_placements[['ssc_p']])

#minimum and maximum of normalized weight:
df_placements['min_max_scaled_ssc_p'].min(),df_placements['min_max_scaled_ssc_p'].max()
```

Out[9]: (0.0, 1.0)

In [11]:
```
#boxcox is only for large datasets
#reduce the skewness from the dataset
#it is effective than log transform
import scipy
from scipy import stats

#scipy.stats.boxcox( returns tuple with transformation:
#[0] refers to array of the indexing particular columns in displacement
box_displacement = scipy.stats.boxcox(df_placements['ssc_p'])

#distribution of transformed variable:
sns.distplot(box_displacement[0])

#set axes label:
plt.ylabel('ssc_p')
plt.xlabel('gender')

plt.show()
```

```
C:\Users\Zia\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [16]:
```
#get the count of missing values:
missing_values = df_placements.isnull().sum()

#check for missing values:
total = df_placements.isnull().sum().sort_values(ascending = False)

#calculate the percentage of null values:
percent = ((df_placements.isnull().sum()/df_placements.shape[0])*100)

#sort the values in descending order
percent = percent.sort_values(ascending = False)

#concatenate the total missing values and percntage of the missing values:
missing_data = pd.concat([total,percent],axis =1,
                         keys = ['Total','percentage'])

#add the data type:
missing_data['Type'] = df_placements[missing_data.index].dtypes

#view the missing data
missing_data
```

Out[16]:

	Total	percentage	Type
salary	67	31.162791	float64
sl_no	0	0.000000	int64
workex	0	0.000000	object
Scaled_ssc_p	0	0.000000	float64
status	0	0.000000	object
mba_p	0	0.000000	float64
specialisation	0	0.000000	object
etest_p	0	0.000000	float64
degree_t	0	0.000000	object
gender	0	0.000000	object
degree_p	0	0.000000	float64
hsc_s	0	0.000000	object
hsc_b	0	0.000000	object
hsc_p	0	0.000000	float64
ssc_b	0	0.000000	object
ssc_p	0	0.000000	float64
min_max_scaled_ssc_p	0	0.000000	float64

In [25]:
```
me = df_placements['salary'].median()
me
```

Out[25]: 265000.0

In [26]:
```
df_placements['salary'].fillna(value=me,inplace=True)
```

In [27]:
```
#sanity check:
missing_values = df_placements["salary"].isnull().sum()
missing_values
```

Out[27]: 0

In [28]:
```
#filtering the numerical column from the dataset
df_num =df_placements.select_dtypes(include = [np.number])
```

In [29]:
```
df_num
```

Out[29]:

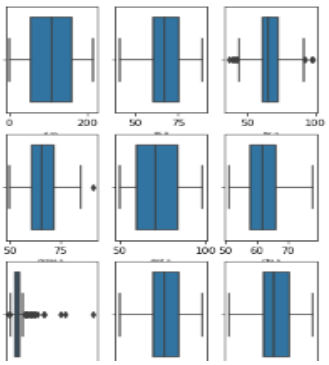
	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary	Scaled_ssc_p	min_max_scaled_ssc_p
0	1	67.00	91.00	58.00	55.0	58.80	270000.0	-0.028087	0.538240
1	2	79.33	78.33	77.48	86.5	66.28	200000.0	1.113369	0.792414
2	3	65.00	68.00	64.00	75.0	57.80	250000.0	-0.213238	0.497011
3	4	56.00	52.00	52.00	66.0	59.43	265000.0	-1.046417	0.311482
4	5	85.80	73.60	73.30	96.8	55.50	425000.0	1.712332	0.925788
...	...	...	...	...	...	...	...	...	...
210	211	80.60	82.00	77.60	91.0	74.49	400000.0	1.230940	0.818594
211	212	58.00	60.00	72.00	74.0	53.62	275000.0	-0.861266	0.352711
212	213	67.00	67.00	73.00	59.0	69.72	295000.0	-0.028087	0.538240
213	214	74.00	66.00	58.00	70.0	60.23	204000.0	0.619941	0.682540
214	215	62.00	58.00	53.00	89.0	60.22	265000.0	-0.490964	0.435168

215 rows × 9 columns

In [39]:
```
df_placements.columns
```

Out[39]:
```
Index(['sl_no', 'gender', 'ssc_p', 'ssc_b', 'hsc_p', 'hsc_b', 'hsc_s',
       'degree_p', 'degree_t', 'workex', 'etest_p', 'specialisation', 'mba_p',
       'status', 'salary', 'Scaled_ssc_p', 'min_max_scaled_ssc_p'],
      dtype='object')
```

In [40]:
```
#plot the boxplot for each columns:
#subplots():plot the subplots
#figsize(): set the figure size
fig, ax = plt.subplots(3,3,figsize = (5,8))

#plot the boxplot using boxplot() from seaborn library:
#z: Let the variable z defines the boxplot
#x : data from which the boxplot is to be predicted
#orient : 'h'  specifies the horizontal boxplot
#whisker : the line form the lower quartile and minimum and upper quartile to maximum ,the default value is 3 in max=1.5 and max
#Inter quartile range(IQR): it is the  method  usedto reduce the outliers
#in IQR we minus the min value and add max value
#whisker is the proportion of the IQR passed from the low and high quartile to extend the plot whiskers
#for upper limit we have to add and for lower limit we have to subtract
#z-score: -3<val<3
#when your dataset does not follow normal distribution yoyu have to follow IQR method to remove the outlies from the dataset.
#when your dataset follows normal distribution you have to follow z-score method to remove the outliers form the dataset
#ax : specifies the axes
# Set_xlabel(): ste the x-axis label
#FonrSize(): set the font size of the x-axis
for variable , subplot in zip(df_num.columns, ax.flatten()):
    #flatten: x and y axes should be flattened
    z = sns.boxplot(x = df_num[variable], orient ='h', whis = 1.5,ax = subplot)
    z.set_xlabel(variable,fontsize = 5)
```



Based on iqr method:

In [41]:
```
#obtain the First Quartile:
Q1 = df_num.quantile(0.25)

#obtain the Quartile:
Q3 = df_num.quantile(0.75)

#obtain the IQR:
IQR = Q3-Q1

# print the IQR:
print(IQR)
```

```
sl_no                  107.000000
ssc_p                   15.100000
hsc_p                   12.100000
degree_p                11.000000
etest_p                 23.500000
mba_p                    8.310000
salary               32500.000000
Scaled_ssc_p             1.397890
min_max_scaled_ssc_p     0.311276
dtype: float64
```

In [43]:
```
#filter out the Outlier values:
# ~: select all the rows which does not satisifies the condition
# any(): returns whether the element is true over the column
# axis =1: indicates should select the alternate columns ('0' for index positions)
from warnings import filterwarnings
filterwarnings('ignore')
df_placements_iqr = df_placements[~((df_placements < (Q1- 1.5 * IQR))| (df_placements > (Q3+1.5 * IQR))).any(axis = 1)]
```

In [44]:
```
df_placements_iqr.shape
```

Out[44]:
```
(175, 17)
```

In [45]:
```
df_placements.shape
```

Out[45]:
```
(215, 17)
```

In [46]:
```
#import the library:
import scipy

#from scipy import Stats module:
from scipy import stats

#z-score are defined for each observation in a avariable
#compute the Z-scores using the method z-score from the scipy library
z_scores_salary = scipy.stats.zscore(df_num['salary'])

#displays the z-score
z_scores_salary
```

Out[46]:
```
0     -0.144570
1     -1.041427
2     -0.400815
3     -0.208631
4      1.841328
         ...
210    1.521022
211   -0.080509
212    0.175736
213   -0.990178
214   -0.208631
Name: salary, Length: 215, dtype: float64
```

In [47]:
```
#printing the rows where the z-score is less than -3
row_index_less = np.where(z_scores_salary< - 3)

print(row_index_less)
```

```
(array([], dtype=int64),)
```

In [50]:
```
#printing the rows where the z-score is less than 3
row_index_more = np.where(z_scores_salary> 3)

print(row_index_more)
```

```
(array([119, 150, 177], dtype=int64),)
```

In [51]:
```
#count of outliers in the variable representing profit:

len(row_index_less[0]) + len(row_index_more[0])
```

Out[51]:
```
3
```

In [53]:
```
#filter out the outlier values :
# ~:select all the rows which do not satisify the condition

df_Zscore_salary = df_placements['salary'][~((z_scores_salary < -3)|(z_scores_salary> 3))]
```

In [55]:
```
#check for the shape:
df_Zscore_salary.shape
```

Out[55]:
```
(212,)
```

Supervised Learning:

In [80]:
```
data = pd.read_csv('WineQT.csv')
data
```

Out[80]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	0
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5	1
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5	2
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6	3
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6	1592
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82	9.5	6	1593
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5	1594
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6	1595
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5	1597

1143 rows × 13 columns

In [81]:
```
data.isnull().sum()
```

Out[81]:
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
```

In [82]:
```
from sklearn.preprocessing import StandardScaler

ss_train = StandardScaler()
X_train = ss_train.fit_transform(X_train)

ss_test = StandardScaler()
X_test = ss_test.fit_transform(X_test)
```

In [83]:
```
#Load library of Test Train and Split
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.25, random_state = 100)
print("X_train:",x_train.shape)
print("X_test:",x_test.shape)
print("Y_train:",y_train.shape)
print("Y_test:",y_test.shape)
```

```
X_train: (426, 30)
X_test: (143, 30)
Y_train: (426,)
Y_test: (143,)
```

In [84]:
```
from sklearn.linear_model import LogisticRegression
```

In [85]:
```
#Create an instance:
le = LogisticRegression()
model = le.fit(X, Y)
```

```
In [84]: from sklearn.linear_model import LogisticRegression

In [85]: #Create an instance:
         le = LogisticRegression()
         model = le.fit(X, Y)

In [86]: le

Out[86]: LogisticRegression()

In [87]: predict = model.predict(X_test)

In [88]: #Load confusion matrix
         from sklearn.metrics import confusion_matrix

         cm = confusion_matrix(y_test, predictions)
         #ravel(): it will fill an empty sets automatically
         TN, FP, FN, TP = confusion_matrix(y_test, predict).ravel()


         print("True positives(TP)=", TP)
         print("False positives(FP)=", FP)
         print("True Negative(TN)=", TN)
         print("False positives(FP)=", FP)

         True positives(TP)= 84
         False positives(FP)= 3
         True Negative(TN)= 53
         False positives(FP)= 3
```

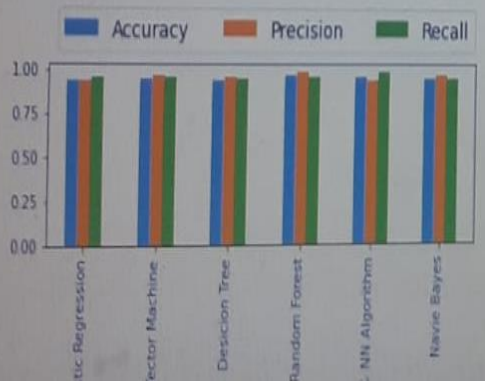
```
In [89]: # accuracy:
         accuracy =(TP+TN) / ( TP+ TN +TN+FN)

         print("Accuracy of the binary classifier:{:0.3f}".format(accuracy))

         Accuracy of the binary classifier:0.710

In [90]: models = {}

         #Logistic regression:
         from sklearn.linear_model import LogisticRegression

         models['Logistic Regression'] = LogisticRegression()

         #support Vector Machines:
         from sklearn.svm import LinearSVC
         models['Support Vector Machine'] = LinearSVC()


         #Desicion Tree:
         from sklearn.tree import DecisionTreeClassifier
         models['Desicion Tree'] = DecisionTreeClassifier()

         #random  forest:
         from sklearn.ensemble import RandomForestClassifier
         models['Random Forest'] = RandomForestClassifier()

         # k-Nearest Neighbouts:
         from sklearn.neighbors import KNeighborsClassifier
         models['K- NN Algorithm'] = KNeighborsClassifier()

         #Navie bayes classifier:
         from sklearn.naive_bayes import GaussianNB
         models['Navie Bayes'] = GaussianNB()
```

```
[91]: from sklearn.metrics import accuracy_score, precision_score, recall_score

      #creating an instance
      accuracy,precision,recall = {},{},{}

      for key in models.keys():
          # fit the classifier:
          models[key].fit(X_train,y_train)

          #make prediction:
          predictions = models[key].predict(X_test)

          #calculate metrics:
          accuracy[key] = accuracy_score(predictions, y_test)
          precision[key] = precision_score(predictions, y_test)
          recall[key] = recall_score(predictions,y_test)
```

```
[92]: import pandas as pd

      df_model = pd.DataFrame(index = models.keys(),
                              columns = ['Accuracy','Precision','Recall'])

      df_model['Accuracy'] = accuracy.values()
      df_model['Precision'] = precision.values()
      df_model['Recall'] = recall.values()

      df_model
```

	Accuracy	Precision	Recall
Logistic Regression	0.937063	0.942529	0.953488
Support Vector Machine	0.951049	0.965517	0.954545
Desicion Tree	0.937063	0.954023	0.943182
Random Forest	0.965035	0.988506	0.955556
K- NN Algorithm	0.958042	0.942529	0.987952
Navie Bayes	0.944056	0.965517	0.943820

```
In [93]: #visualization of score:
         ax = df_model.plot.bar()
         ax.legend( ncol= len(models.keys()),
                 bbox_to_anchor = (0, 1),
                 loc = 'lower left',
                 prop = {'size':14})

         plt.tight_layout()
```



Unsupervised Learning:

In [2]:
```
wine = pd.read_csv('wine.xls')
wine
```

Out[2]:

	1	14.23	1.71	2.43	15.6	127	2.8	3.06	.28	2.29	5.64	1.04	3.92	1065
0	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
1	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
2	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
3	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735
4	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
172	3	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.70	0.64	1.74	740
173	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750
174	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835
175	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840
176	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560

177 rows × 14 columns

In [3]:
```
wine.dtypes
```

In [7]:
```
# changing the correct column names:
wine.head()
```

Out[7]:

	1	14.23	1.71	2.43	15.6	127	2.8	3.06	.28	2.29	5.64	1.04	3.92	1065
0	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
1	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
2	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
3	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735
4	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450

In [42]:
```
wine_tra.rename(columns={'1':'Alcohol','14.23':'Malic Acid','1.71':'Ash','2.43':'Alkalinity od Ash','15.6':'Malic acid ash','127'
```

In [43]:
```
wine_tra
```

Out[43]:

	Alcohol	Malic Acid	Ash	Alkalinity od Ash	Malic acid ash	Total Phenol	Flavanoid	Nonflavoured Phenols	Proanthyocyanins	Color Intensity	Hue	OD218/OD315 of diluted wines	Proline	Flavoured ethanol	label
0	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050	1
1	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185	1
2	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480	1
3	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735	0

In [42]:
```
wine_tra.rename(columns={'1':'Alcohol','14.23':'Malic Acid','1.71':'Ash','2.43':'Alkalinity od Ash','15.6':'Malic acid ash','127'
```

In [43]:
```
wine_tra
```

Out[43]:

	Alcohol	Malic Acid	Ash	Alkalinity od Ash	Malic acid ash	Total Phenol	Flavanoid	Nonflavoured Phenols	Proanthyocyanins	Color Intensity	Hue	OD218/OD315 of diluted wines	Proline	Flavoured ethanol	label
0	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050	1
1	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185	1
2	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480	1
3	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735	0
4	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
171	3	14.16	2.51	2.48	20.0	91	1.68	0.70	0.44	1.24	9.70	0.62	1.71	660	0
173	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750	0
174	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835	0
175	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840	0
176	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560	2

In [10]:
```
#excluding the target column:

x = wine_tra.drop(['Alcohol'], axis = 1)
y = wine_tra['Alcohol']
```

In [11]:
```
#import the train_test_split module from sklearn
from sklearn.model_selection import train_test_split
#Let us split the dataset into test and train
#Test Size: The proportion of the Data to be included in the testing set.

x_train, x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2, random_state = 100)

print("X_train:",x_train.shape)
print("X_test:",x_test.shape)
print("Y_train:",y_train.shape)
print("Y_test:",y_test.shape)
```

```
X_train: (128, 13)
X_test: (33, 13)
Y_train: (128,)
Y_test: (33,)
```

K- Means Algorithm:

In [15]:
```
from sklearn.preprocessing import StandardScaler

#creating an Instance
ss= StandardScaler()
wine_ss = ss.fit_transform(wine_tra)
wine_ss = pd.DataFrame(wine_tra)
wine_ss.head()
```

Out[15]:

	Alcohol	Malic Acid	Ash	Alkalinity od Ash	Malic acid ash	Total Phenol	Flavanoid	Nonflavoured Phenols	Proanthyocyanins	Color Intensity	Hue	OD218/OD315 of diluted wines	Proline	Flavoured ethanol
0	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
1	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
2	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
3	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735
4	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450

In [23]:
```
from sklearn.cluster import AgglomerativeClustering,KMeans
```

K- Means Algorithm:

In [15]:
```
from sklearn.preprocessing import StandardScaler

#creating an Instance
ss= StandardScaler()
wine_ss = ss.fit_transform(wine_tra)
wine_ss = pd.DataFrame(wine_tra)
wine_ss.head()
```

Out[15]:

	Alcohol	Malic Acid	Ash	Alkalinity od Ash	Malic acid ash	Total Phenol	Flavanoid	Nonflavoured Phenols	Proanthyocyanins	Color Intensity	Hue	OD218/OD315 of diluted wines	Proline	Flavoured ethanol
0	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
1	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
2	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
3	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735
4	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450

In [23]:
```
from sklearn.cluster import AgglomerativeClustering,KMeans
#finding the optimal value of k:
err =[]
for i in range(1,10):
    km = KMeans(n_clusters = i, random_state = 100)
```

In [23]:
```
from sklearn.cluster import AgglomerativeClustering,KMeans
#finding the optimal value of k:
err =[]
for i in range(1,10):
    km = KMeans(n_clusters = i, random_state = 100)
    km.fit(wine_ss)
    err.append(km.inertia_)#inertia_ is nothing but centroid value
```

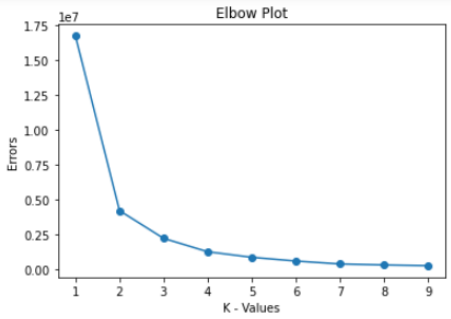
In [24]:
```
err
```

Out[24]:
```
[16718761.05505439,
 4184615.4683733955,
 2196253.5006604404,
 1237349.0622886084,
 837344.2731759606,
 570841.4176542228,
 360934.9458883859,
 288622.0047835235,
 237810.3457378806]
```

In [25]:
```
#sanity check
plt.plot(range(1,10), err ,marker ='o')
plt.xlabel('K - Values')
plt.ylabel("Errors")
plt.title("Elbow Plot")
plt.show()
```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code



In [32]:
```
#finding the optimal k value using silhoutte score:

from sklearn.metrics import silhouette_score

for i in range(2,10):
    km2 = KMeans(n_clusters = i, random_state = 100)
    km2.fit(wine_ss)
    sil_score = silhouette_score(wine_ss, km2.labels_)
    print("The silhoutte_score for",i,"K - value is",sil_score)
```

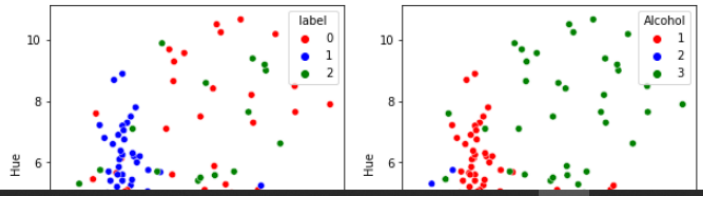
```
The silhoutte_score for 2 K - value is 0.6598073119999515
The silhoutte_score for 3 K - value is 0.5725993557212941
The silhoutte_score for 4 K - value is 0.561342291791319
The silhoutte_score for 5 K - value is 0.5589119298474148
The silhoutte_score for 6 K - value is 0.5784330394787254
The silhoutte_score for 7 K - value is 0.5718495794895878
The silhoutte_score for 8 K - value is 0.5453904286213305
The silhoutte_score for 9 K - value is 0.5353275566122231
```

In [37]:
```
#visualise the score:
from yellowbrick.cluster import SilhouetteVisualizer


for i in range(2,10):
    km2 = KMeans(n_clusters = i, random_state = 100)
    km2.fit(wine_ss)
    sil_score= silhouette_score(wine_ss, km2.labels_)
    print("Silhouette score for",i,"k - value is",sil_score)
    sil_km2 = SilhouetteVisualizer(km2)
    sil_km2.fit(wine_ss)
    plt.show()
    plt.tight_layout()
```

In [ ]:
```
# no of categoreis= no of cluters (value count of particular variable = no of clusters)
```

In [38]:
```
km3 = KMeans(n_clusters =3,random_state = 100)
km3.fit(wine_ss)
y_pred_km = km3.predict(wine_ss)
wine_tra['label'] = y_pred_km
```

In [44]:
```
#plot our columns:
plt.figure(figsize = (10,5))
plt.subplot(1,2,1)
sns.scatterplot(wine_tra['Ash'],wine_tra['Hue'],hue = wine_tra['label'], palette = ['red','blue','green'])
plt.subplot(1,2,2)
sns.scatterplot(wine_tra['Ash'],wine_tra['Hue'], hue = wine_tra['Alcohol'], palette = ['red','blue','green'])

plt.show()
```



In [ ]:
```
#selecting optimal linkage method:

link = ['single','complete','average','centroid','ward']
for i in link:
    z = linkage(wine_ss,i)
```

In [ ]:
```
ag =AgglomerativeClustering(n_clusters = 3, linkage =)
```

In [ ]:
```
wine_tra['label_ag'] = y_pred_ag
```

In [ ]:
```
plt.figure(figsize = (7,7))
plt.subplot(1,2,1)
sns.scatterplot(wine_tra['Ash'],wine_tra['Hue'],hue = wine_tra['label_ag'], palette = ['red','blue','green'])
plt.subplot(1,2,2)
sns.scatterplot(wine_tra['Ash'],wine_tra['Hue'],hue = wine_tra['Alcohol'], palette = ['red','blue','green'])
plt.subplot(1,3,3)
sns.scatterplot(wine_tra['Ash'],wine_tra['Hue'], hue = wine_tra['label'], palette = ['red','blue','green'])

plt.show()
```

In [ ]:
```
tuning your accuracy score is called hyperparameter tuning
```

In [45]:
```
#hyperparameter tuning for logistic regression:
model1 = LogisticRegression(C=100,max_iter =100)
from sklearn.model_selection import GridSearchCV

params =[{'C':[1,5,10]},{'max_iter':[100,50]}]

#we chose precision scoring since it is a classification health domain dataset:
model = GridSearchCV(model1,param_grid = params,scoring = 'precision', cv = 5) #param_grid is a keyword
model1_grid = model.fit(X,Y)

model.fit(x_train, y_train)

print(model.best_params_)
```

FINAL PROJECT:ONLINE RESTAURANT RATE PREDICTION

online restaurant rating

importiing libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#import plotly.plotly as py
#import plotly.graph_objs as go

#py.offline.init_notebook_mode(connected=True)

#%matplotlib notebook
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

In [6]:
```
# import data set
data = pd.read_csv("C:/Users/MONIKA/Downloads/Untitled Folder/zomato.csv")
data
```

Out[6]:

	url	address	name	online_order	book_table	rate	votes	
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	42
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	+9
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	+9
4	https://www.zomato.com/bangalore/grand-village	10, 3rd Floor, Lakshmi Associates	Grand Village	No	No	3.8/5	166	8026

data preprocessing

```
data.isna().sum()
```

```
url                                0
address                            0
name                               0
online_order                       0
book_table                         0
rate                            7775
votes                              0
phone                           1208
location                          21
rest_type                        227
dish_liked                     28078
cuisines                          45
approx_cost(for two people)      346
reviews_list                       0
menu_item                          0
listed_in(type)                    0
listed_in(city)                    0
dtype: int64
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 17 columns):
url                            51717 non-null object
address                        51717 non-null object
name                           51717 non-null object
online_order                   51717 non-null object
book_table                     51717 non-null object
rate                           43942 non-null object
votes                          51717 non-null int64
phone                          50509 non-null object
location                       51696 non-null object
rest_type                      51490 non-null object
dish_liked                     23639 non-null object
cuisines                       51672 non-null object
approx_cost(for two people)    51371 non-null object
reviews_list                   51717 non-null object
menu_item                      51717 non-null object
listed_in(type)                51717 non-null object
listed_in(city)                51717 non-null object
dtypes: int64(1), object(16)
memory usage: 6.7+ MB
```

In [8]:
```
data=data[data.cuisines.isna()==False]
```

In [9]:
```
data.isna().sum()
```

Out[9]:
```
url                                0
address                            0
name                               0
online_order                       0
book_table                         0
rate                            7741
votes                              0
phone                           1179
location                           0
rest_type                        206
dish_liked                     28033
cuisines                           0
approx_cost(for two people)      320
reviews_list                       0
menu_item                          0
listed_in(type)                    0
listed_in(city)                    0
dtype: int64
```

In [11]:
```
data.drop(columns=["url", 'address','phone','listed_in(city)'], inplace  =True)
```

In [12]:
```
data.rename(columns={'approx_cost(for two people)': 'average_cost'}, inplace=True)
```

In [13]:
```
data.rename(columns={'listed_in(type)': 'listed_type'}, inplace=True)
```

In [14]:
```
data.name.value_counts().head()
```

Out[14]:
```
Cafe Coffee Day      96
Onesta               85
Just Bake            73
Empire Restaurant    71
Five Star Chicken    70
Name: name, dtype: int64
```

In [16]:
```
data.online_order.value_counts()
```

Out[16]:
```
Yes    30428
No     21244
Name: online_order, dtype: int64
```

data visualization

In [17]:
```
ax= sns.countplot(data['online_order'])
plt.title('Number of Restaurants accepting online orders', weight='bold')
plt.xlabel('online orders')
```

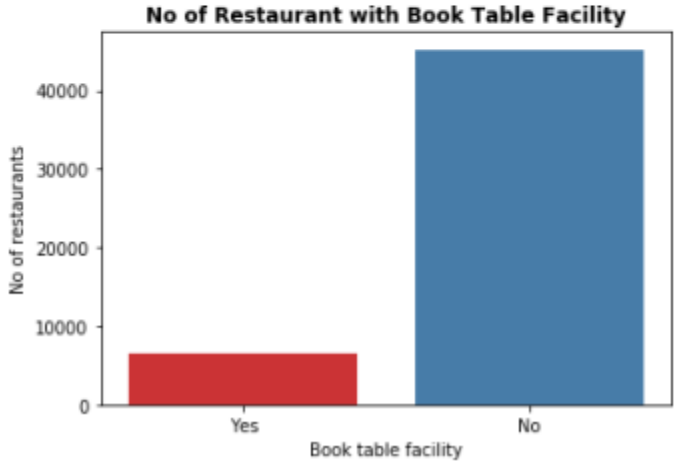
Out[17]:
```
Text(0.5, 0, 'online orders')
```

In [18]:
```
data['book_table'].value_counts()
```

Out[18]:
```
No     45223
Yes     6449
Name: book_table, dtype: int64
```

In [19]:
```
sns.countplot(data['book_table'], palette= "Set1")
plt.title("No of Restaurant with Book Table Facility", weight = 'bold')
plt.xlabel('Book table facility')
plt.ylabel('No of restaurants')
```

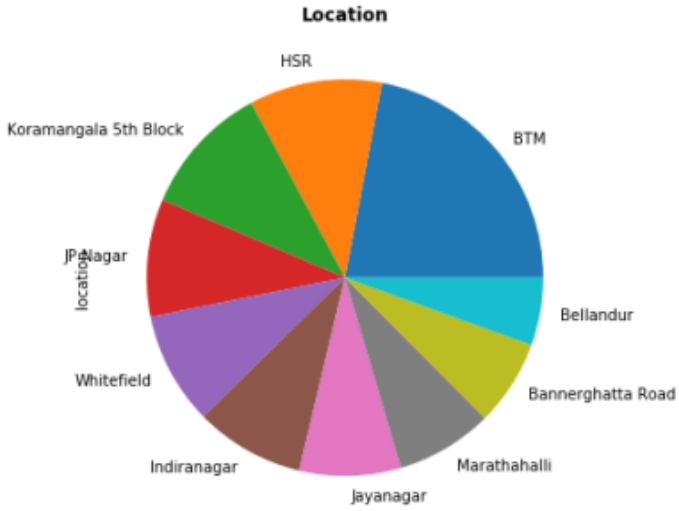
Out[19]:
```
Text(0, 0.5, 'No of restaurants')
```



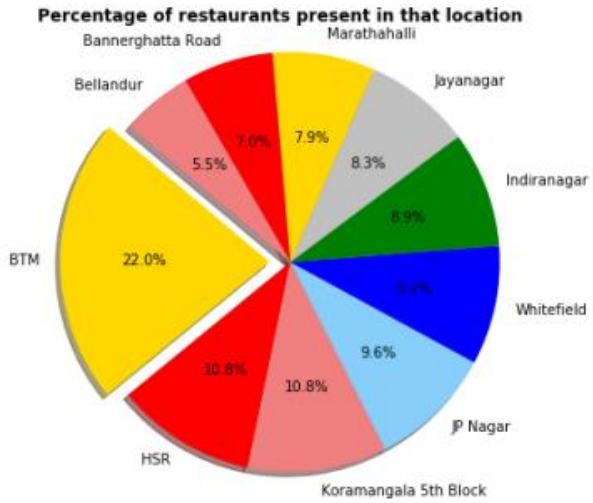
Pie chart

In [21]:
```
plt.figure(figsize=(12,6))
data['location'].value_counts()[:10].plot(kind = 'pie')
plt.title('Location', weight = 'bold')
```

Out[21]: Text(0.5, 1.0, 'Location')



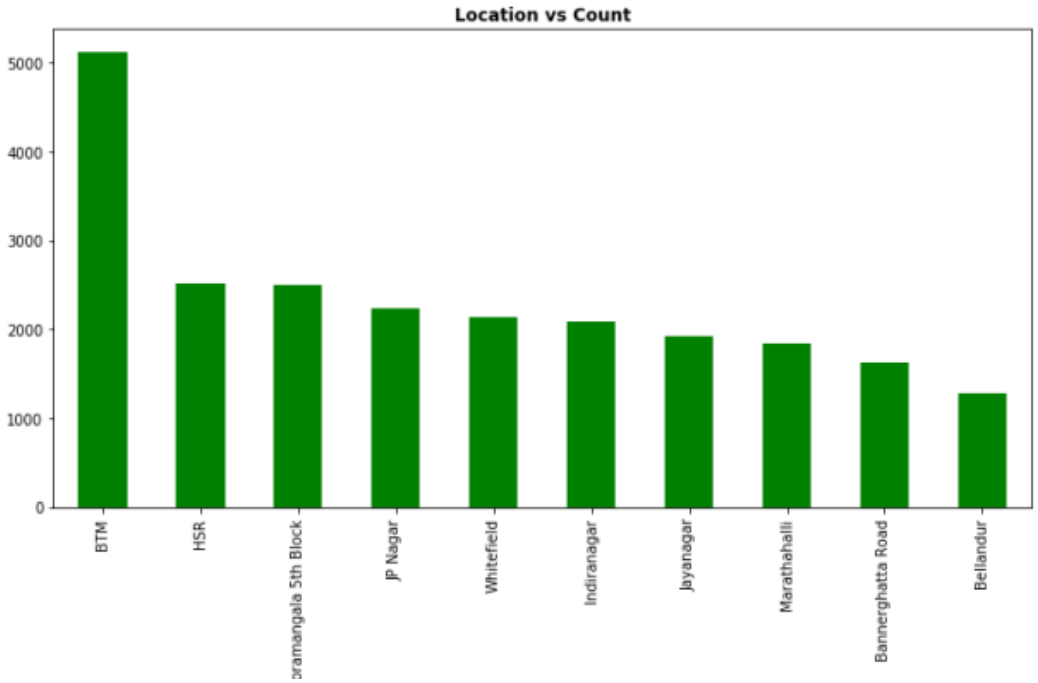
In [22]:
```
plt.figure(figsize = (12,6))
names = data['location'].value_counts()[:10].index
values = data['location'].value_counts()[:10].values
colors = ['gold', 'red', 'lightcoral', 'lightskyblue','blue','green','silver']
explode = (0.1, 0, 0, 0,0,0,0,0,0,0)  # explode 1st slice

plt.pie(values, explode=explode, labels=names, colors=colors,autopct='%1.1f%%', shadow=True, star
plt.axis('equal')
plt.title("Percentage of restaurants present in that location", weight = 'bold')
plt.show()
```



BAR CHART:

In [23]:
```
plt.figure(figsize = (12,6))
data['location'].value_counts()[:10].plot(kind = 'bar', color = 'g')
plt.title("Location vs Count", weight = 'bold')
```

Out[23]: Text(0.5, 1.0, 'Location vs Count')



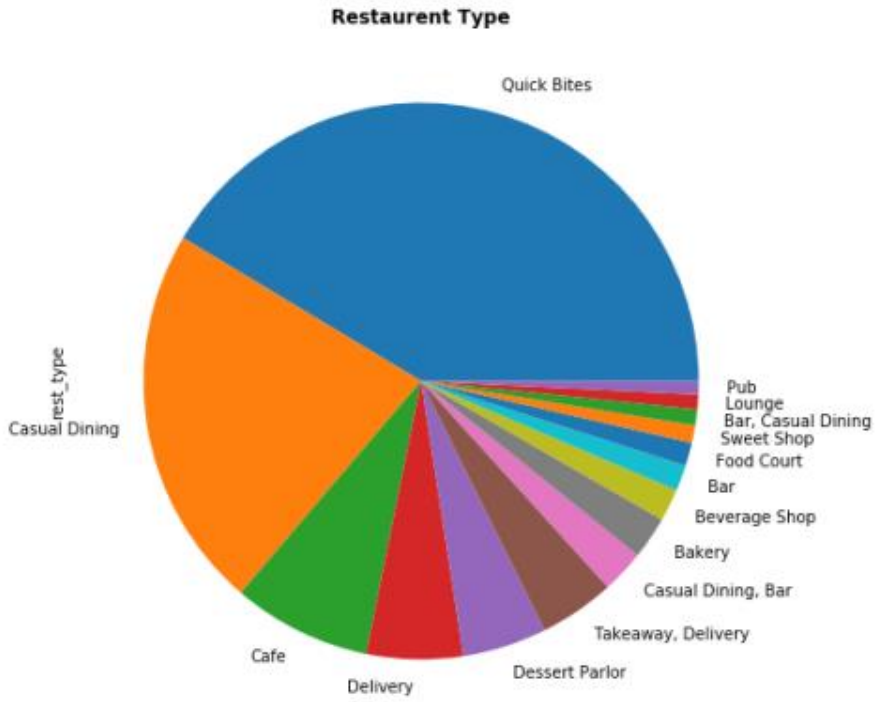
In [24]:
```
data['location'].nunique()
```

Out[24]: 93

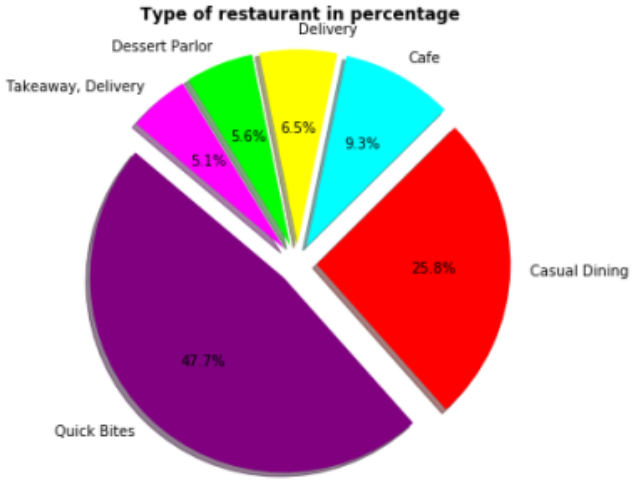
In [25]:
```
data['rest_type'].value_counts().head(10)
```

Out[25]:
```
Quick Bites           19129
Casual Dining         10326
Cafe                   3732
Delivery               2595
Dessert Parlor         2262
Takeaway, Delivery     2035
Casual Dining, Bar     1154
Bakery                 1141
Beverage Shop           865
Bar                     697
Name: rest_type, dtype: int64
```

In [26]:
```
plt.figure(figsize = (14,8))
data.rest_type.value_counts()[:15].plot(kind = 'pie')
plt.title('Restaurent Type', weight = 'bold')
plt.show()
```



In [28]:
```
plt.figure(figsize = (12,6))
names = data['rest_type'].value_counts()[:6].index
values = data['rest_type'].value_counts()[:6].values
explode = (0.1, 0.1, 0.1, 0.1,0.1,0.1)  # explode 1st slice

plt.title('Type of restaurant in percentage', weight = 'bold')
plt.pie(values, explode=explode, labels=names, colors=colors,autopct='%1.1f%%', shadow=True, star
plt.axis('equal')
plt.show()
```



In [31]:
```
colors  = ("red", "green", "orange", "cyan", "brown", "grey", "blue", "indigo", "beige", "yellow"
```

In [33]:
```
dishes_data = data[data.dish_liked.notnull()]
dishes_data.dish_liked = dishes_data.dish_liked.apply(lambda x:x.lower().strip())
```

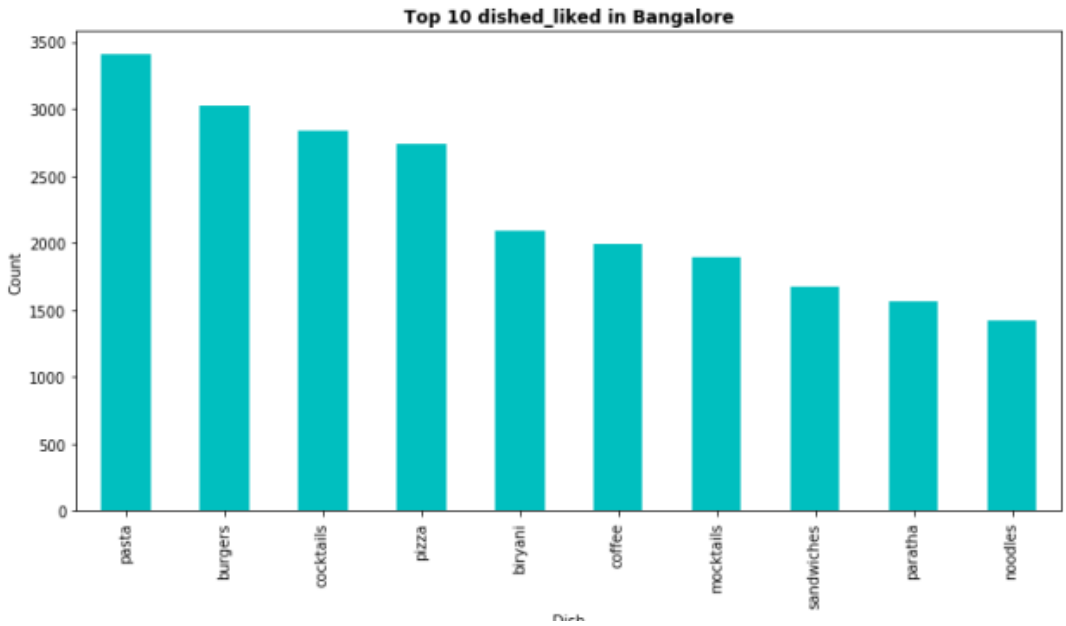
In [34]:
```
dishes_data.isnull().sum()
```

Out[34]:
```
name              0
online_order      0
book_table        0
rate             30
votes             0
location          0
rest_type        70
dish_liked        0
cuisines          0
average_cost    136
reviews_list      0
menu_item         0
listed_type       0
dtype: int64
```

In [35]:
```
# count each dish to see how many times each dish repeated
dish_count = []
for i in dishes_data.dish_liked:
    for t in i.split(','):
        t = t.strip() # remove the white spaces to get accurate results
        dish_count.append(t)
```

In [36]:
```
plt.figure(figsize=(12,6))
pd.Series(dish_count).value_counts()[:10].plot(kind='bar',color= 'c')
plt.title('Top 10 dished_liked in Bangalore',weight='bold')
plt.xlabel('Dish')
plt.ylabel('Count')
```

Out[36]:
```
Text(0, 0.5, 'Count')
```

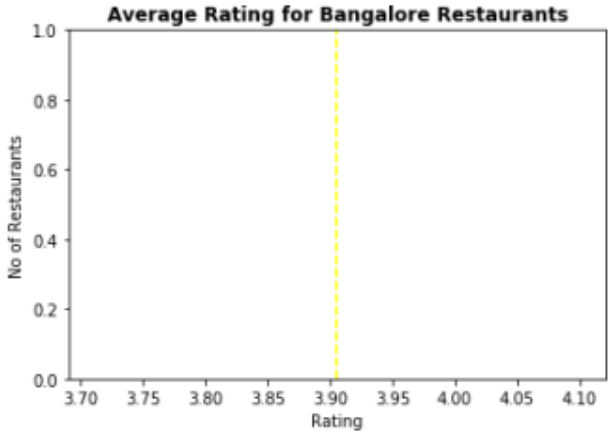


In [37]:
```
data['rate'] = data['rate'].replace('NEW',np.NaN)
data['rate'] = data['rate'].replace('-',np.NaN)
data.dropna(how = 'any', inplace = True)
```

In [38]:
```
data['rate'] = data.loc[:,'rate'].replace('[ ]','',regex = True)
data['rate'] = data['rate'].astype(str)
data['rate'] = data['rate'].apply(lambda r: r.replace('/5',''))
data['rate'] = data['rate'].apply(lambda r: float(r))
```

In [39]:
```
plt.axvline(x= data.rate.mean(),ls='--',color='yellow')
plt.title('Average Rating for Bangalore Restaurants',weight='bold')
plt.xlabel('Rating')
plt.ylabel('No of Restaurants')
print(data.rate.mean())
```

```
3.9058343007007914
```



In [40]:
```
data['online_order']= pd.get_dummies(data.online_order, drop_first=True)
data['book_table']= pd.get_dummies(data.book_table, drop_first=True)
data
```

Out[40]:

	name	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisines	average_cost
0	Jalsa	1	1	4.1	775	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	800
1	Spice Elephant	1	0	4.1	787	Banashankari	Casual Dining	Momos, Lunch Buffet, Chocolate Nirvana, Thai G...	Chinese, North Indian, Thai	800
2	San Churro Cafe	1	0	3.8	918	Banashankari	Cafe, Casual Dining	Churros, Cannelloni, Minestrone Soup, Hot Choc...	Cafe, Mexican, Italian	800
3	Addhuri Udupi Bhojana	0	0	3.7	88	Banashankari	Quick Bites	Masala Dosa	South Indian, North Indian	300
4	Grand Village	0	0	3.8	166	Basavanagudi	Casual Dining	Panipuri, Gol Gappe	North Indian, Rajasthani	600
...	...	...	...	...	...	...	...	...	...	...
51705	Izakaya Gastro Pub	1	1	3.8	128	Whitefield	Bar, Casual Dining	Beer, Chicken Guntur, Paneer Tikka	North Indian, Continental, Mediterranean	1,200

In [41]:
```
data.drop(columns=['dish_liked','reviews_list','menu_item','listed_type'], inplace  =True)
```

In [42]:
```
data['rest_type'] = data['rest_type'].str.replace(',' , '')
data['rest_type'] = data['rest_type'].astype(str).apply(lambda x: ' '.join(sorted(x.split())))
data['rest_type'].value_counts().head()
```

Out[42]:
```
Casual Dining        7331
Bites Quick          5253
Cafe                 2375
Bar Casual Dining    1321
Dessert Parlor       1083
Name: rest_type, dtype: int64
```

In [43]:
```
data['rest_type'] = data['rest_type'].str.replace(',' , '')
data['rest_type'] = data['rest_type'].astype(str).apply(lambda x: ' '.join(sorted(x.split())))
data['rest_type'].value_counts().head()
```

Out[43]:
```
Casual Dining        7331
Bites Quick          5253
Cafe                 2375
Bar Casual Dining    1321
Dessert Parlor       1083
Name: rest_type, dtype: int64
```

In [45]:
```
data['cuisines'] = data['cuisines'].str.replace(',' , '')
data['cuisines'] = data['cuisines'].astype(str).apply(lambda x: ' '.join(sorted(x.split())))
data['cuisines'].value_counts().head()
```

label encoding

46]:
```
from sklearn.preprocessing import LabelEncoder
T = LabelEncoder()
data['location'] = T.fit_transform(data['location'])
data['rest_type'] = T.fit_transform(data['rest_type'])
data['cuisines'] = T.fit_transform(data['cuisines'])
#data['dish_liked'] = T.fit_transform(data['dish_liked'].
```

48]:
```
data["average_cost"] = data["average_cost"].str.replace(',' , '')
```

49]:
```
data["average_cost"] = data["average_cost"].astype('float')
```

50]:
```
data.head()
```

50]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	average_cost
0	Jalsa	1	1	4.1	775	1	29	951	800.0
1	Spice Elephant	1	0	4.1	787	1	29	963	800.0
2	San Churro Cafe	1	0	3.8	918	1	22	806	800.0
3	Addhuri Udupi Bhojana	0	0	3.7	88	1	19	1201	300.0
4	Grand Village	0	0	3.8	166	4	29	1237	600.0

In [51]:
```
x = data.drop(['rate','name'],axis = 1)
```

In [52]:
```
y = data['rate']
```

In [53]:
```
x.shape
```

Out[53]: (23259, 7)

In [54]:
```
y.shape
```

Out[54]: (23259,)

dat training and testing

In [55]:
```
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 = 33)
```

In [56]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23259 entries, 0 to 51715
Data columns (total 9 columns):
name            23259 non-null object
online_order    23259 non-null uint8
book_table      23259 non-null uint8
rate            23259 non-null float64
votes           23259 non-null int64
location        23259 non-null int32
rest_type       23259 non-null int32
cuisines        23259 non-null int32
average_cost    23259 non-null float64
```

In [57]:
```
#standarizing
#taking numeric values
from sklearn.preprocessing import StandardScaler
num_values1=data.select_dtypes(['float64','int64']).columns
scaler = StandardScaler()
scaler.fit(data[num_values1])
data[num_values1]=scaler.transform(data[num_values1])
```

In [58]:
```
data.head()
```

Out[58]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	average_cost
0	Jalsa	1	1	0.455722	0.152328	1	29	951	0.089176
1	Spice Elephant	1	0	0.455722	0.163105	1	29	963	0.089176
2	San Churro Cafe	1	0	-0.248401	0.280757	1	22	806	0.089176
3	Addhuri Udupi Bhojana	0	0	-0.483109	-0.464668	1	19	1201	-0.871467
4	Grand Village	0	0	-0.248401	-0.394616	4	29	1237	-0.295081

linear regression

In [59]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
y_pred_lr = lr.predict(X_test)
```

In [60]:
```
lr.score(X_test, y_test)*100
```

Out[60]:
```
21.319197295401814
```

In [61]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred_lr))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred_lr))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred_lr)))
```

```
Mean Absolute Error: 0.2653603458127496
Mean Squared Error: 0.13773462897554362
Root Mean Squared Error: 0.3711261631514863
```

random forest

In [62]:
```
from sklearn import metrics
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(X_train,y_train)
y_pred_rfr = rfr.predict(X_test)
```

In [63]:
```
rfr.score(X_test,y_test)*100
```

Out[63]:
```
90.99018454023702
```

In [64]:
```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred_rfr))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred_rfr))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred_rfr)))
```

```
Mean Absolute Error: 0.0466433200730223
Mean Squared Error: 0.015772126704752698
Root Mean Squared Error: 0.1255871279421291
```

4.STUDENT FEEDBACK

I am very thankful to you SURE TRUST for providing this wonderful opportunity to me to learn this PYTHON & MACHINE LEARNING course .In this course I learned from basics to deep .This course is very helpful to me not only for me but all of us. I know just basics of python but this trainer (Bhuvanesh sir) make us good at in this course he taught me from the basic level with best examples and solutions He gave a lot of assignments and material for learning and doing projects and coding parts. Heartful thanks to Radha Kumari madam and their management for providing this course to me with free of cost. Now I can clearly and confidently say that I can perform good research and obtain formal information and data on any topic, as opposed to just surfing the internet for genuine knowledge.

5.DISTINCTIVENESS OF THE COURSE

- Concepts are learnt theoretically and practically.
- Faculty is committed in delivering the course.
- Through out the course best assignments will be given.
- Preparation of the course report helps us to refer in the future.
- Trainer teaching is easy to understand for us.
- All students can do projects because of the trainer will cared about all the students.
- Asking questions in between the sessions helps us students for better understanding.
- Clarifying doubts in that session only.
- By doing Assignments we can learn easily.
- Hands on practice experience.

6.CONCLUDING REMARKS

A lot of experience, knowledge and exposure will be gained. All disclosures were awaken myself in a boost of self-confidence to face life more challenging now. Practical knowledge is a complement to the science rather than theoretical knowledge. I conclude that the SURE TRUST training program has provided many benefits to the students. Its my pleasure to be a student of SURE TRUST.

BY

SURE TRUST STUDENT

MONIKA GANGA KARRI