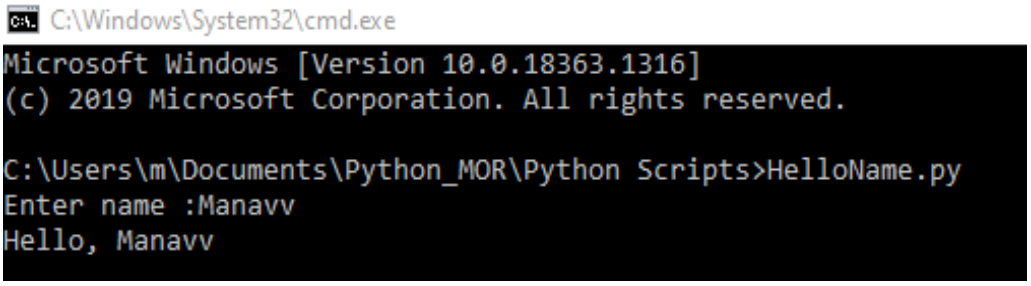
PYTHON PROGRAMMING (MOR)
MANAVV KALRA
SECTION : B
HANSRAJ COLLEGE

1) Write a program to enter name and display as “Hello, Name”

```
def main():
  name= input("Enter name :")
  print("Hello,",name)

if __name__=="__main__":
  main()
```

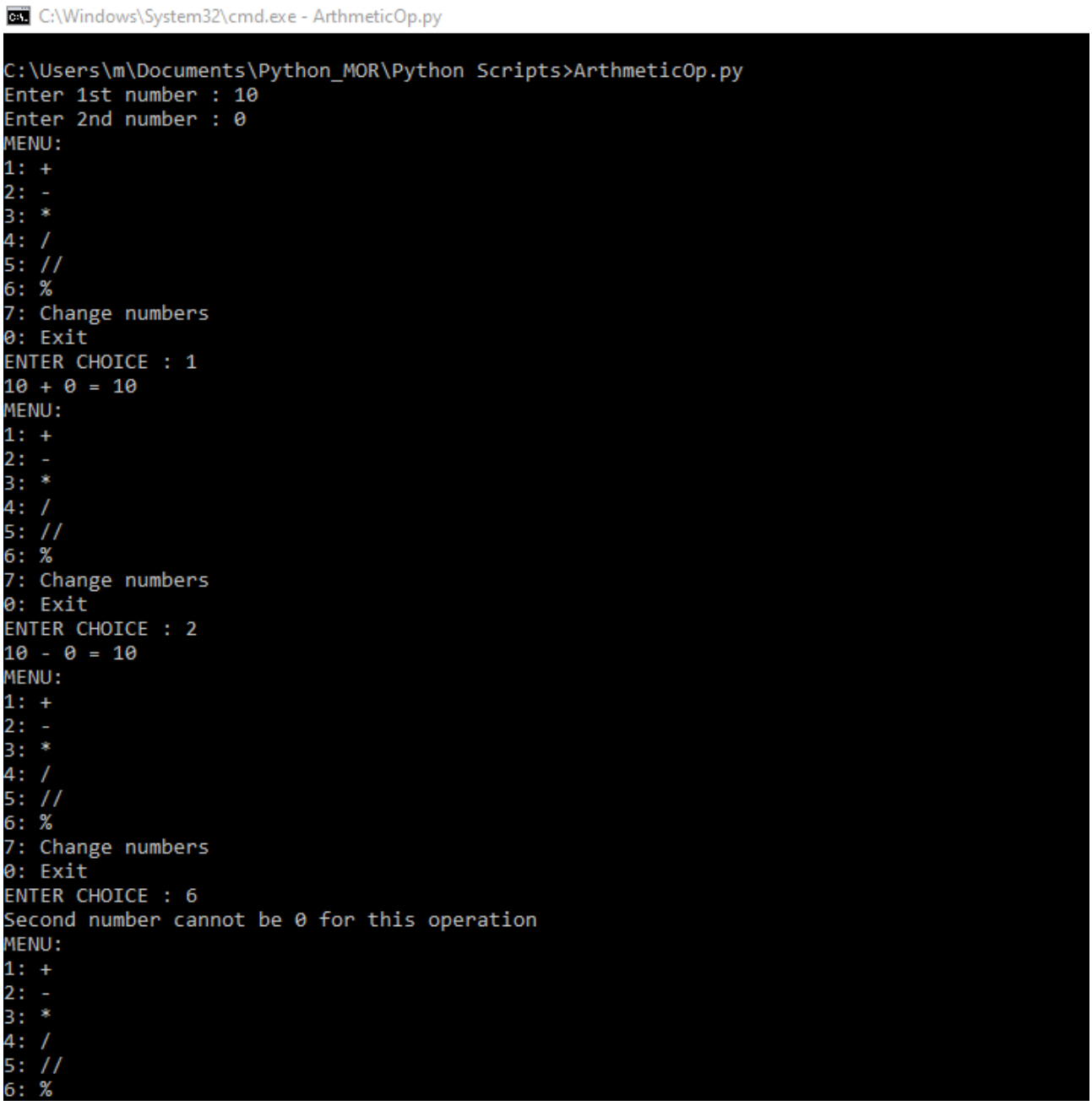
OUTPUT



2) Write a menu driven program to enter two numbers and print the arithmetic operations like a.+ b.– c.* d./ e. // f. %.

```
def main():
  x=int(input("Enter 1st number : "))
  y=int(input("Enter 2nd number : "))
  while(1):
    print("MENU: \n1: +\n2: - \n3: *\n4: / \n5: // \n6: %\n7: Change numbers\n0: Ex
it")
    char = int(input("ENTER CHOICE : "))
    if(char==1):
      print(x,"+",y,"=",x+y)
    elif(char==2):
      print(x,"-",y,"=",x-y)
    elif(char==3):
      print(x,"*",y,"=",x*y)
    elif(char==4):
      if(y==0):
        print("Second number cannot be 0 for this operation")
      else:
        print(x,"/",y,"=",x/y)
    elif(char==5):
      if(y==0):
        print("Second number cannot be 0 for this operation")
      else:
        print(x,"//",y,"=",x//y)
    elif(char==6):
      if(y==0):
        print("Second number cannot be 0 for this operation")
      else:
        print(x,"%",y,"=",x%y)
    elif(char==7):
      main()
    elif(char==0):
      exit()
    else :
      print("INVALID CHOICE")


if __name__=="__main__":
  main()
```

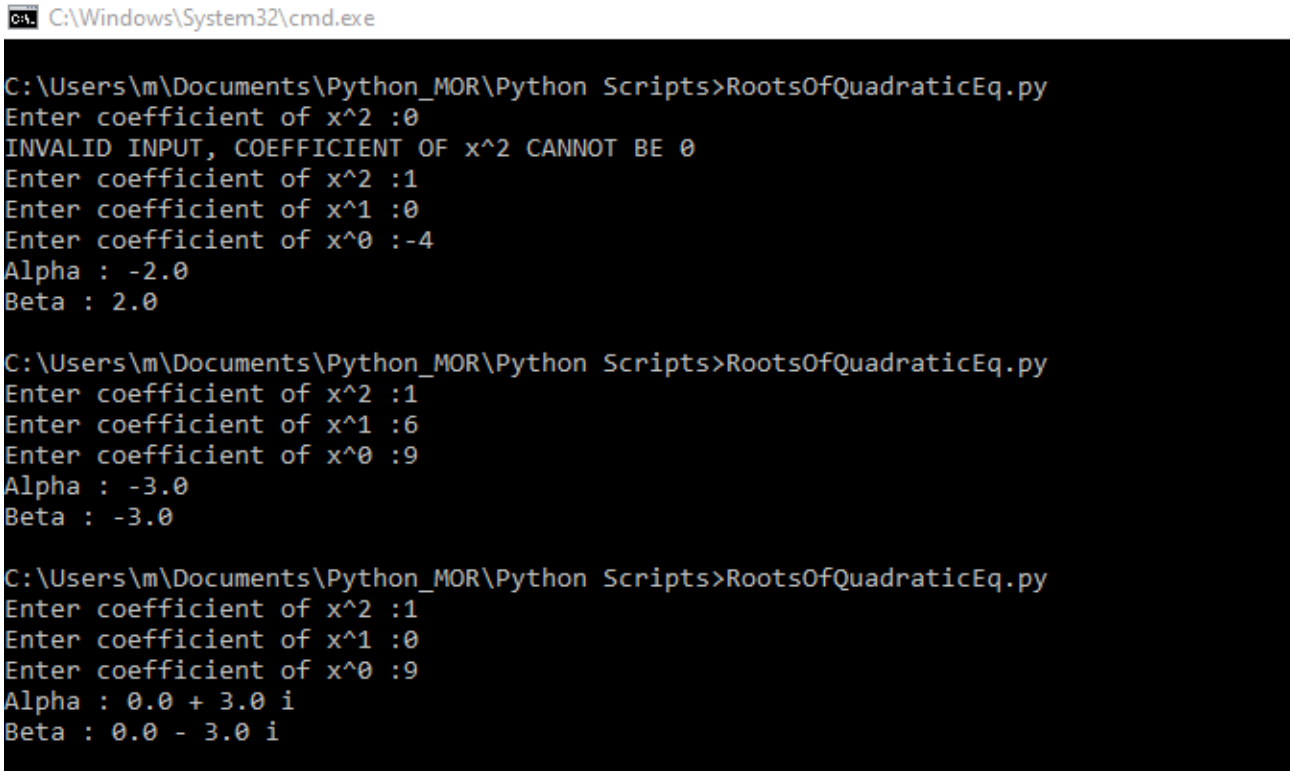
OUTPUT


C:\Windows\System32\cmd.exe - ArthmeticOp.py
C:\Users\m\Documents\Python_MOR\Python Scripts>ArthmeticOp.py
Enter 1st number : 10
Enter 2nd number : 0
MENU:
1: +
2: -
3: *
4: /
5: //
6: %
7: Change numbers
0: Exit
ENTER CHOICE : 1
10 + 0 = 10
MENU:
1: +
2: -
3: *
4: /
5: //
6: %
7: Change numbers
0: Exit
ENTER CHOICE : 2
10 - 0 = 10
MENU:
1: +
2: -
3: *
4: /
5: //
6: %
7: Change numbers
0: Exit
ENTER CHOICE : 6
Second number cannot be 0 for this operation
MENU:
1: +
2: -
3: *
4: /
5: //
6: %

3) To compute the roots of a quadratic equation.

```
import math
def compute(a,b,c):
  d = (b**2) - (4*a*c)
  if (d<0):
    r=-b/(2*a)
    im=math.sqrt(-d)/(2*a)
    print("Alpha :",r,"+",im,"i")
    print("Beta :",r,"-",im,"i")
  else:
    alpha = (-b-math.sqrt(d))/(2*a)
    beta = (-b+math.sqrt(d))/(2*a)
    print("Alpha :",alpha)
    print("Beta :",beta)

def main():
  a = int(input("Enter coefficient of x^2 :"))
  if(a==0):
    print("INVALID INPUT, COEFFICIENT OF x^2 CANNOT BE 0")
    main()
  else:
    b = int(input("Enter coefficient of x^1 :"))
    c = int(input("Enter coefficient of x^0 :"))
    compute(a,b,c)

if __name__=="__main__":
  main()
```

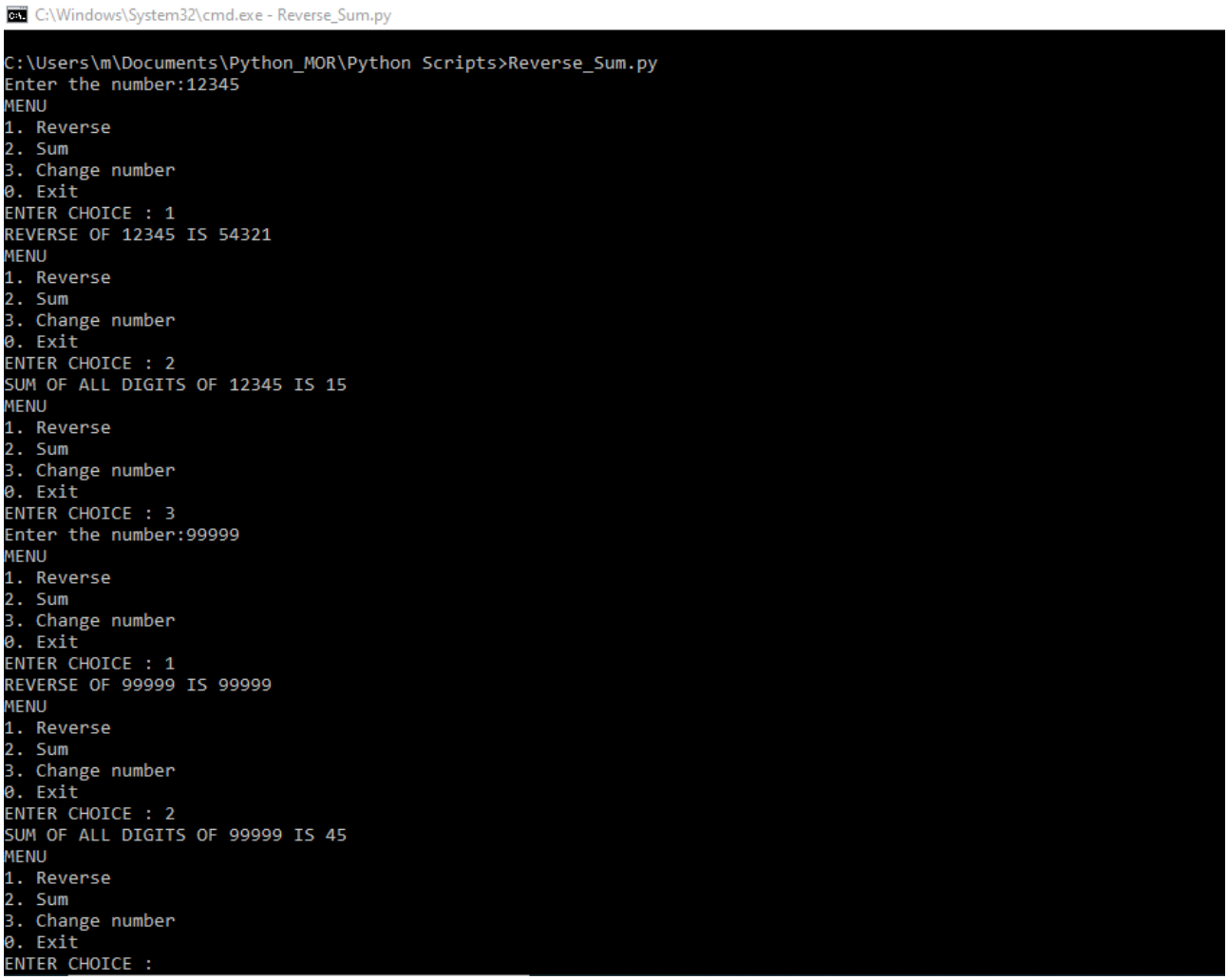
OUTPUT



4) Write a menu driven Program to reverse the entered numbers and print the sum of digits entered.

```
def compute(n):
  rev = 0
  addall = 0
  while(n > 0):
    rem = n%10
    rev = (rev *10) + rem
    addall+=rem
    n=n//10
  return rev,addall

def main():
  n = int(input("Enter the number:"))
  reverse, sum= compute(n)
  while(1):
    print("MENU")
    print("1. Reverse")
    print("2. Sum")
    print("3. Change number")
    print("0. Exit")
    char = int(input("ENTER CHOICE : "))
    if(char==1):
      print("REVERSE OF",n,"IS",reverse)
    elif(char==2):
      print("SUM OF ALL DIGITS OF",n,"IS",sum)
    elif(char==3):
      main()
    elif(char==0):
      exit()
    else :
      print("INVALID CHOICE")


if __name__=="__main__":
  main()
```

OUTPUT

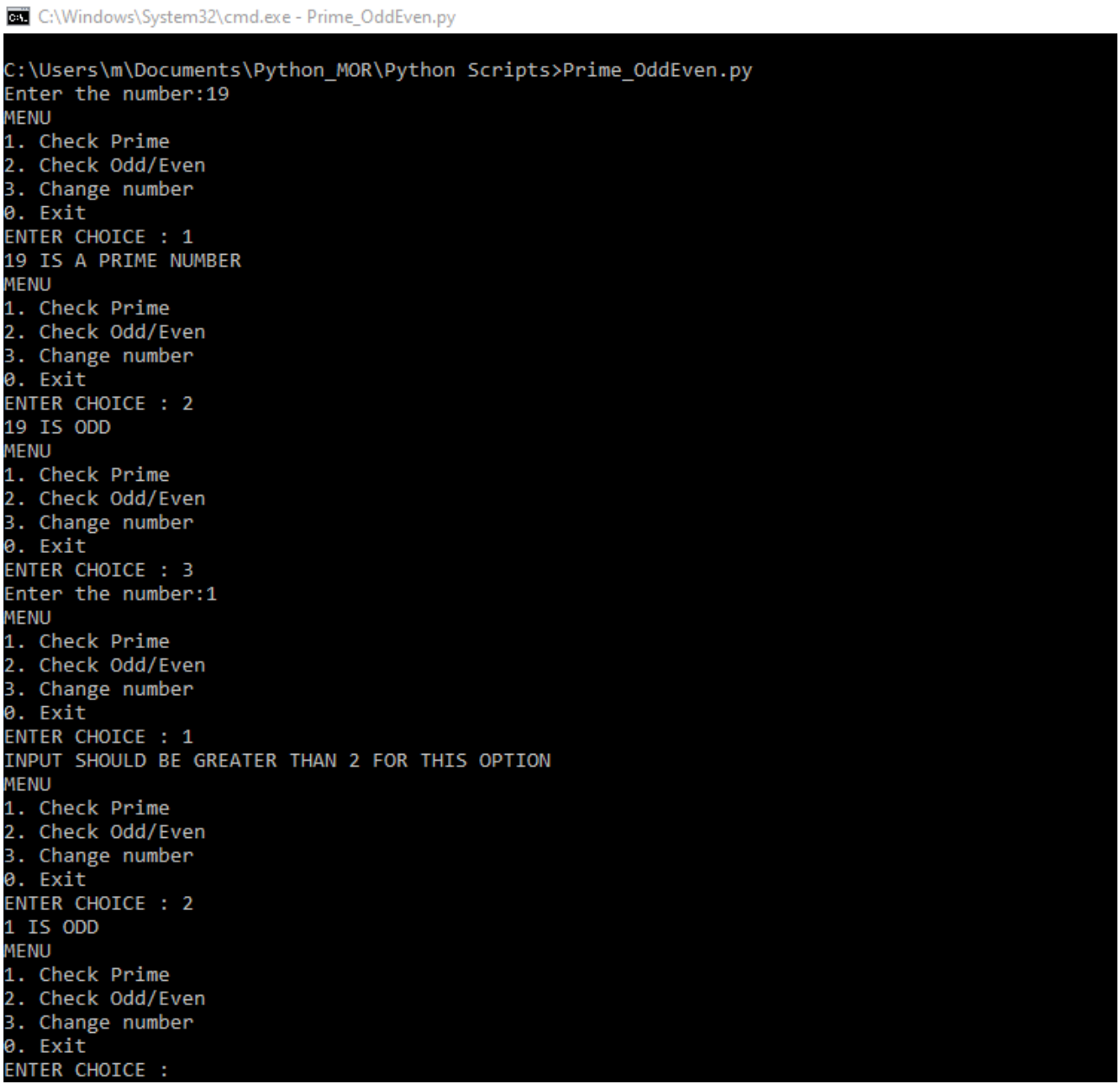

C:\Windows\System32\cmd.exe - Reverse_Sum.py
C:\Users\m\Documents\Python_MOR\Python Scripts>Reverse_Sum.py
Enter the number:12345
MENU
1. Reverse
2. Sum
3. Change number
0. Exit
ENTER CHOICE : 1
REVERSE OF 12345 IS 54321
MENU
1. Reverse
2. Sum
3. Change number
0. Exit
ENTER CHOICE : 2
SUM OF ALL DIGITS OF 12345 IS 15
MENU
1. Reverse
2. Sum
3. Change number
0. Exit
ENTER CHOICE : 3
Enter the number:99999
MENU
1. Reverse
2. Sum
3. Change number
0. Exit
ENTER CHOICE : 1
REVERSE OF 99999 IS 99999
MENU
1. Reverse
2. Sum
3. Change number
0. Exit
ENTER CHOICE : 2
SUM OF ALL DIGITS OF 99999 IS 45
MENU
1. Reverse
2. Sum
3. Change number
0. Exit
ENTER CHOICE :

5) Write menu driven Program to enter the number and print whether the number is
a. Odd or even
b. Prime

```
import math

def checkPrime(n):
  flag = 0
  for i in range(2,math.floor(math.sqrt(n))+1):
    if(n%i==0):
      flag=1
      break
  return flag

def checkOddEven(n):
  flag = 0
  if(n%2==0):
    flag=1
  return flag

def main():
  n = int(input("Enter the number:"))
  while(1):
    print("MENU")
    print("1. Check Prime")
    print("2. Check Odd/Even")
    print("3. Change number")
    print("0. Exit")
    char = int(input("ENTER CHOICE : "))
    if(char==1):
      if(n<2):
        print("INPUT SHOULD BE GREATER THAN 2 FOR THIS OPTION")
      else:
        pc=checkPrime(n)
        if(pc==1):
          print(n,"IS NOT A PRIME NUMBER")
        else:
          print(n,"IS A PRIME NUMBER")
    elif(char==2):
      oc=checkOddEven(n)
      if(oc==1):
        print(n,"IS EVEN")
      else:
        print(n,"IS ODD")
    elif(char==3):
      main()
    elif(char==0):
      exit()
    else :
      print("INVALID CHOICE")
```

```
if __name__=="__main__":
  main()
```

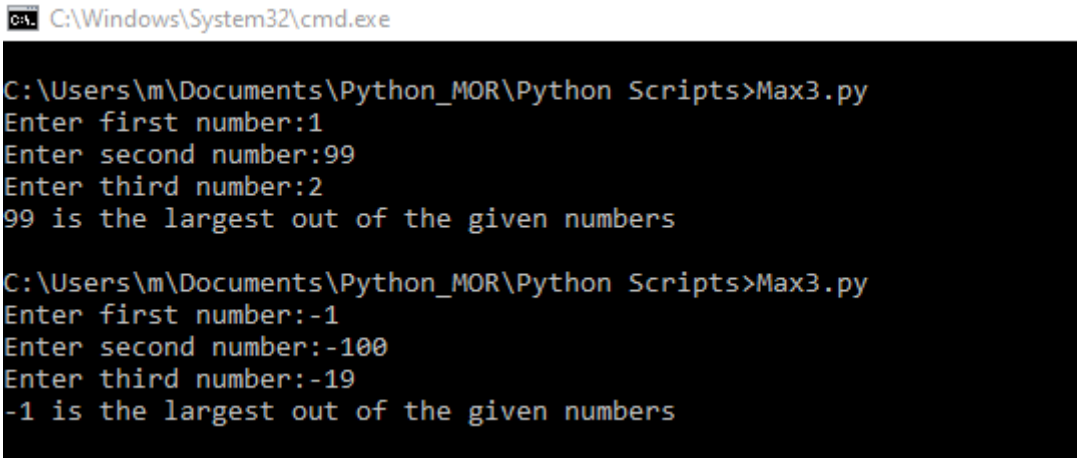
OUTPUT



6) Program to find maximum out of entered 3 numbers

```
def maxout3(x,y,z):
  if (x >= y) and (x >= z):
    max=x
  elif (y >= x) and (y >= z):
    max=y
  else:
    max=z
  return max

def main():
  num1 = int(input("Enter first number:"))
  num2 = int(input("Enter second number:"))
  num3 = int(input("Enter third number:"))
  cm=maxout3(num1,num2,num3)
  print(cm,"is the largest out of the given numbers")

if __name__=="__main__":
  main()
```

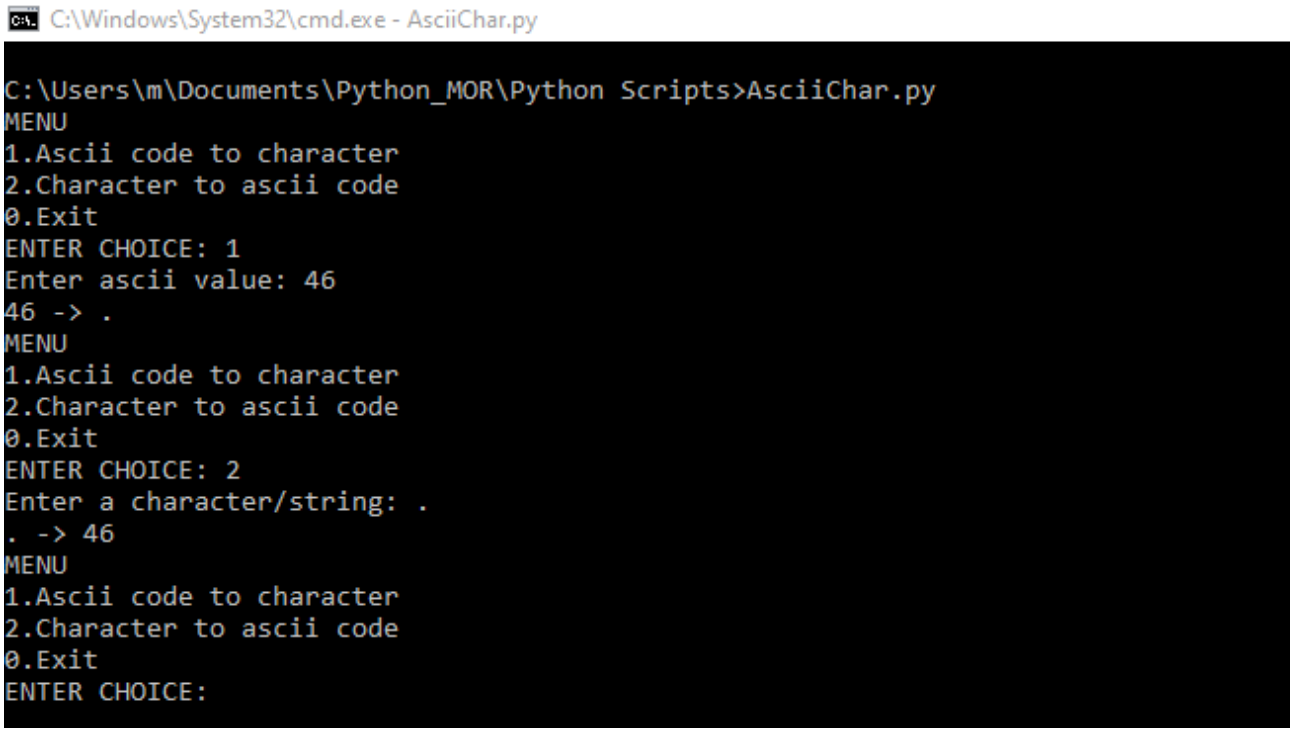
OUTPUT



7) Write a program to display ASCII code of a character and vice versa.

```
def AsciiToChar():
  n=int(input("Enter ascii value: "))
  print(n,"->",chr(n))

def CharToAscii():
   str=input("Enter a character/string: ")
   for i in str:
     print(i,"->",ord(i))

def main():
   while(1):
     print("MENU")
     print("1.Ascii code to character")
     print("2.Character to ascii code")
     print("0.Exit")
     char=int(input("ENTER CHOICE: "))
     if(char==1):
       AsciiToChar()
     elif(char==2):
       CharToAscii()
     elif(char==0):
       exit()
     else :
       print("INVALID CHOICE")

if __name__=="__main__":
  main()
```

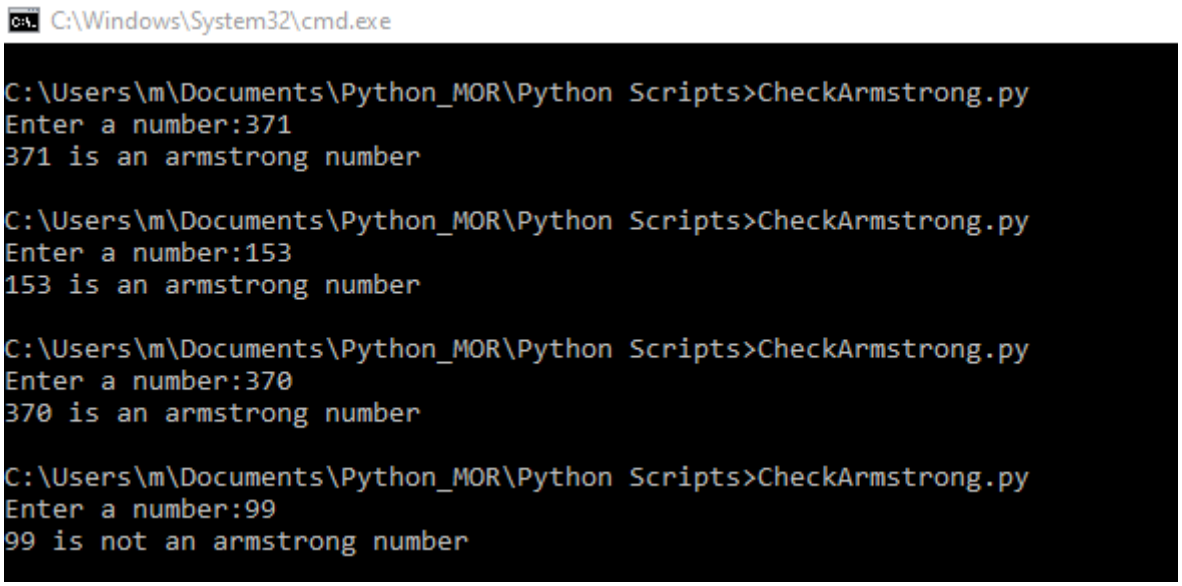
OUTPUT



8) Write a Program to check if the entered number is Armstrong or not.

```
def checkarmstrong(n):
  length=len(n)
  n=int(n)
  dummy=n
  sum=0
  while(dummy>0):
    temp=dummy%10
    sum+=temp**length
    dummy=dummy//10
  if (sum==n):
    return True
  else:
    return False

def main():
  n = input("Enter a number:")
  if (checkarmstrong(n)):
    print(n,"is an armstrong number")
  else:
    print(n,"is not an armstrong number")

if __name__=="__main__":
  main()
```

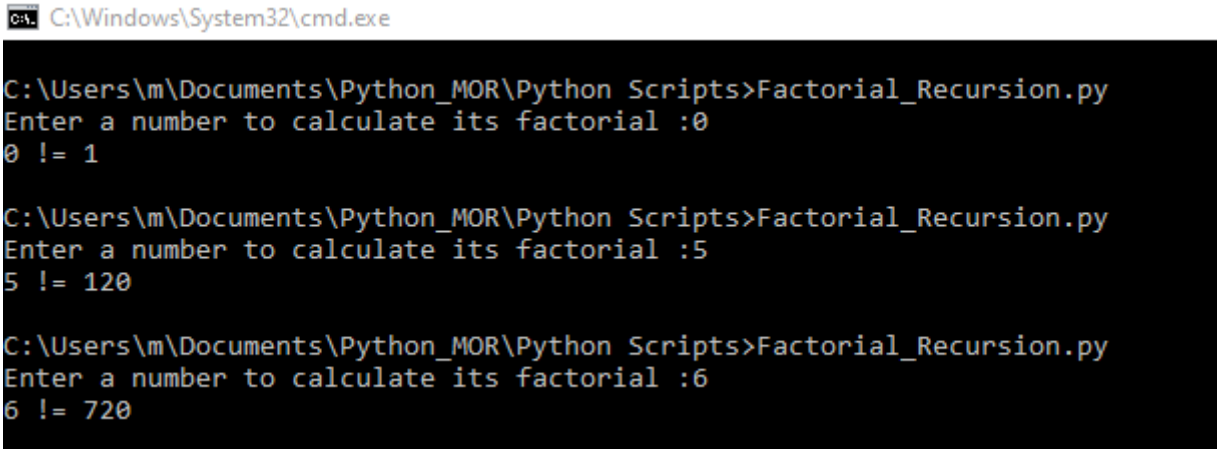
OUTPUT



9) Write a Program to find factorial of the entered number using recursion.

```
def factorial(n):
  if (n==0) or (n == 1):
    return 1
  else:
    return n*factorial(n-1)

def main():
  n = int(input("Enter a number to calculate its factorial :"))
  if(n<0):
    print("INPUT SHOULD NOT BE NEGATIVE")
  else:
    fact=factorial(n)
    print(n,"!=",fact)

if __name__=="__main__":
  main()
```

OUTPUT



10) Write a Program to enter the number of terms and to print the Fibonacci Series.

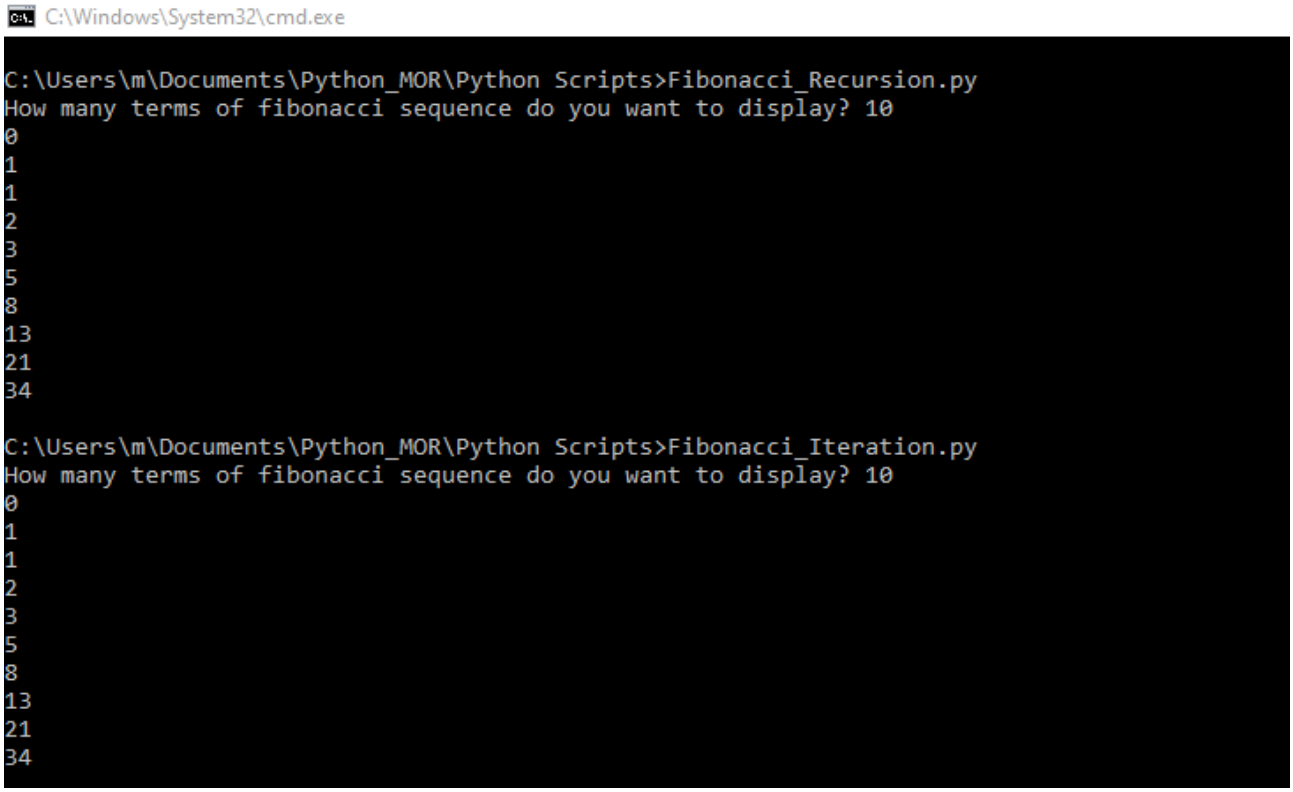
FIBONACCI (ITERATION)

```
def fibonacci(n):
  first=0
  second=1
  for i in range(1,n+1):
    print(first)
    temp=first+second
    first=second
    second=temp


def main():
  n = int(input("How many terms of fibonacci sequence do you want to display? "))
  fibonacci(n)

if __name__=="__main__":
  main()
```

FIBONACCI (RECURSION)

```
def fibonacci(n):
  if (n==1):
    return 0
  elif (n==2):
    return 1
  else:
    return (fibonacci(n-1)+fibonacci(n-2))

def main():
  n = int(input("How many terms of fibonacci sequence do you want to display? "))
  for i in range(1,n+1):
    print(fibonacci(i))

if __name__=="__main__":
  main()
```

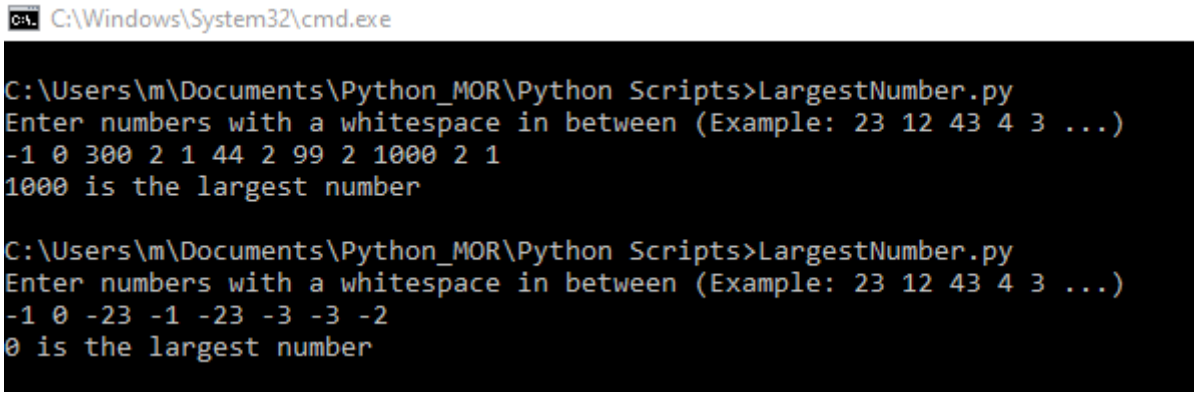
OUTPUT


C:\Windows\System32\cmd.exe
C:\Users\m\Documents\Python_MOR\Python Scripts>Fibonacci_Recursion.py
How many terms of fibonacci sequence do you want to display? 10
0
1
1
2
3
5
8
13
21
34
C:\Users\m\Documents\Python_MOR\Python Scripts>Fibonacci_Iteration.py
How many terms of fibonacci sequence do you want to display? 10
0
1
1
2
3
5
8
13
21
34

11) Write a Program to enter the numbers and to print greatest number using loop.

```
def findMax(arr):
  ini = arr[0]
  for i in arr:
    if i > ini :
      ini = i
  return ini

def main():
  print("Enter numbers with a whitespace in between (Example: 23 12 43 4 3 ...) ")
  arr = [int(x) for x in input().split()]
  print(findMax(arr),"is the largest number")

if __name__=="__main__":
  main()
```

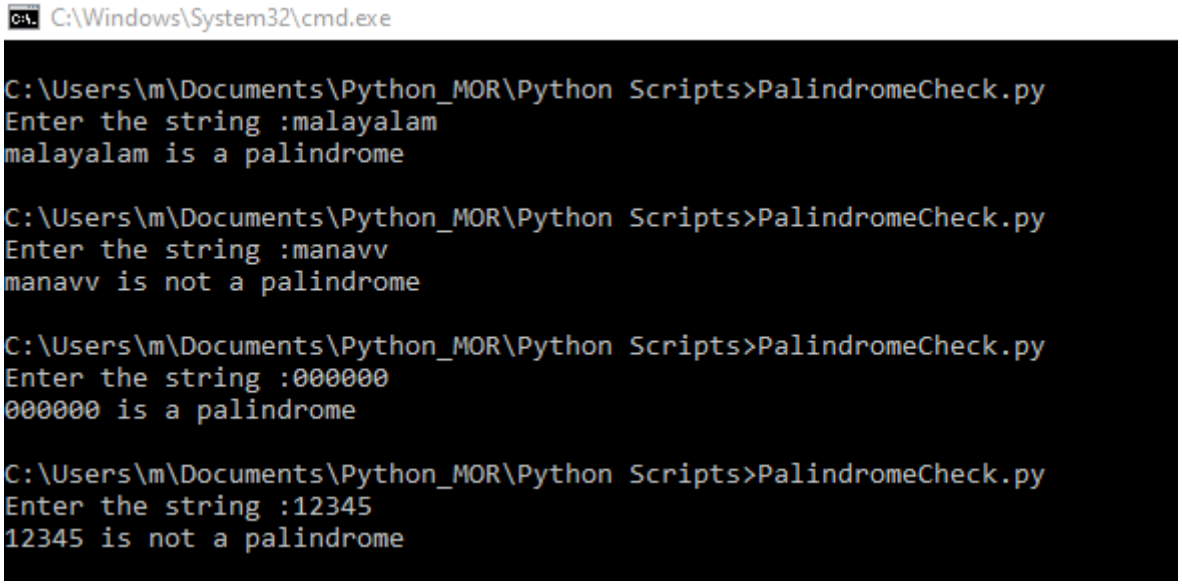
OUTPUT



12) Write a Program to enter the string and to check if it's palindrome or not using loop.

```
def PalindromeCheck(str):
  reverse=""
  for i in str:
    reverse=i+reverse
  return reverse==str

def main():
  str=input("Enter the string :")
  if(PalindromeCheck(str)):
    print(str,"is a palindrome")
  else:
    print(str,"is not a palindrome")

if __name__=="__main__":
  main()
```

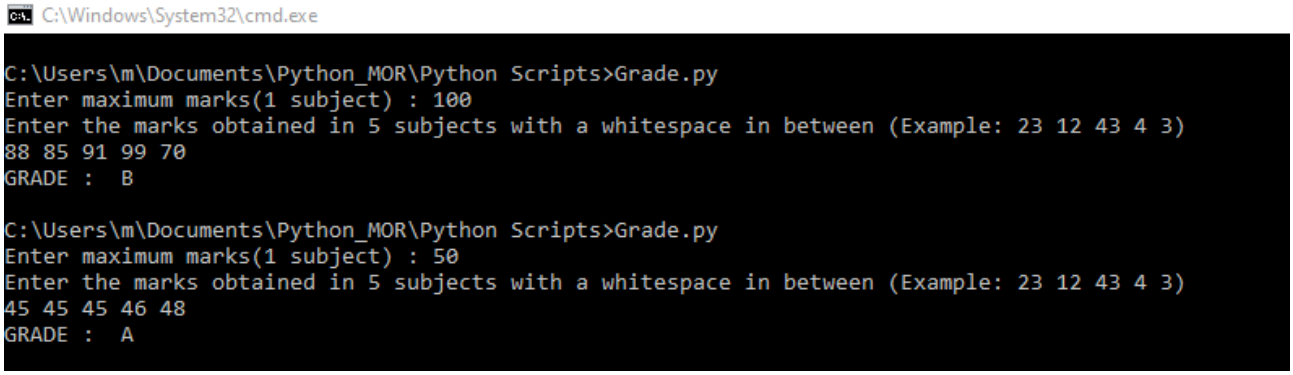
OUTPUT



13) Write a Program to enter the 5 subjects numbers and print the grades A/B/C/D/E.

```
def grade(maxmarks,obtainedmarks):
  percentage=(sum(obtainedmarks)/(5*maxmarks))*100
  if(percentage>=91):
    return "A"
  elif(percentage>=81 and percentage<91):
    return "B"
  elif(percentage>=71 and percentage<81):
    return "C"
  elif(percentage>=61 and percentage<71):
    return "D"
  else:
    return "E"

def main():
  mm=int(input("Enter maximum marks(1 subject) : "))
  if(mm<0):
    print("MAXIMUM MARKS CANNOT BE NEGATIVE")
    main()
  else:
    print("Enter the marks obtained in 5 subjects with a whitespace in between (Exa
mple: 23 12 43 4 3) ")
    arr = [float(x) for x in input().split()]
    if(len(arr)!=5):
      print("ENTER MARKS FOR 5 SUBJECTS")
      main()
    elif(max(arr)>mm):
      print("OBTAINED MARKS CANNOT BE MORE THAN MAXIMUM MARKS")
      main()
    else:
      print("GRADE : ",grade(mm,arr))

if __name__=="__main__":
  main()
```

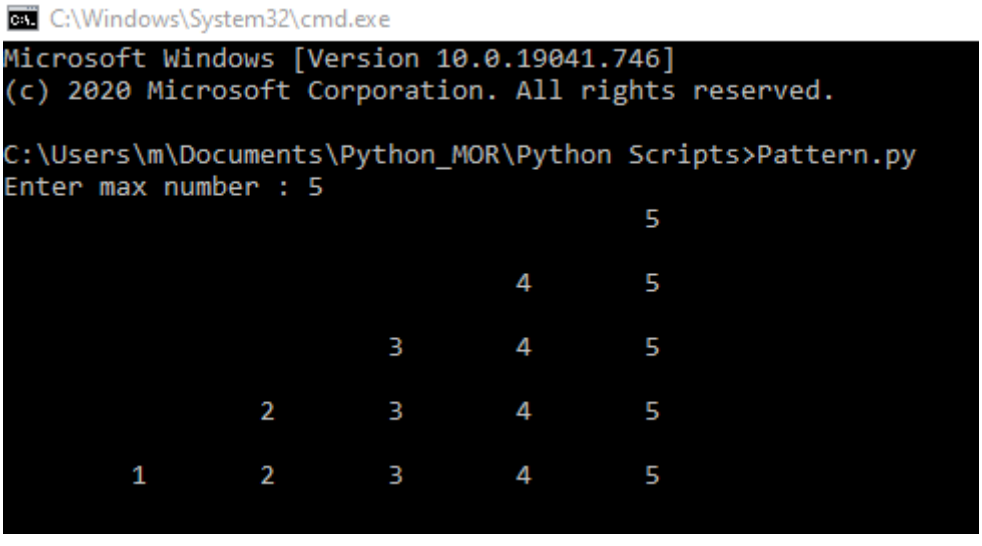
OUTPUT



14) Write a program in python language to display the given pattern :


```
def pattern():
  n=int(input("Enter max number : "))
  k=n-1
  for i in range(n,0,-1):
    print(k*'\t',end="\t")
    for j in range(i,n+1):
      print(j,end="\t")
    k-=1
    print('\n')

if __name__=="__main__":
  pattern()
```

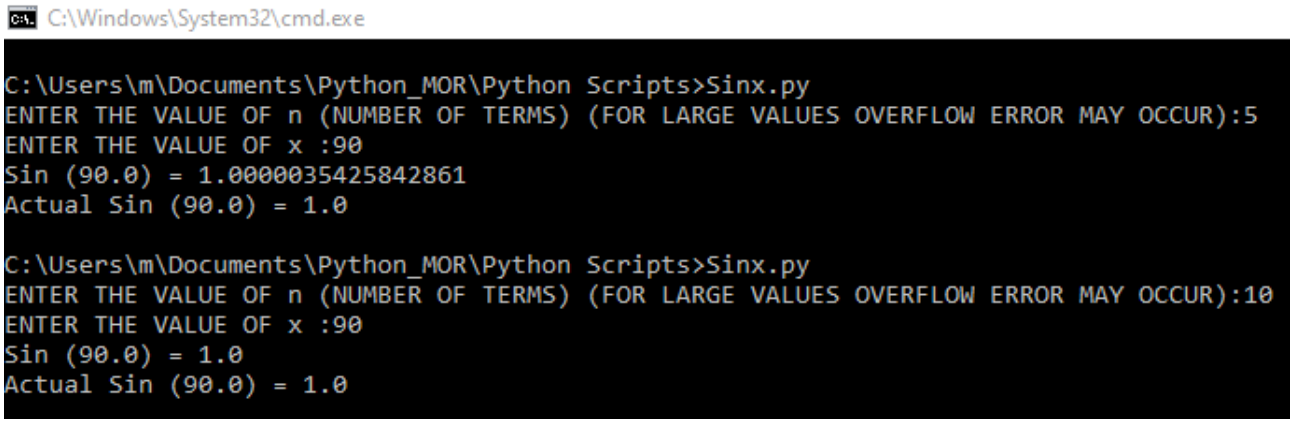
OUTPUT



15) Write a python function sin(x,n) to calculate the value of sin(x) using its Taylor series expansion up to n terms.

```
import math

def sin(x,n):
  sinx=0
  for i in range (n):
    sinx+=((-1)**i)*(x**(2*i+1))/math.factorial(2*i+1)
  return sinx

def main():
  n=int(input("ENTER THE VALUE OF n (NUMBER OF TERMS) (FOR LARGE VALUES OVERFLOW ER
ROR MAY OCCUR):"))
  if(n<1):
    print("n CANNOT BE LESS THAN 1")
    main()
  else:
    x=float(input("ENTER THE VALUE OF x :"))
    print("Sin ({}) = {}".format(x,sin(math.radians(x),n)))
    print("Actual Sin ({}) = {}".format(x,math.sin(math.radians(x))))

if __name__=="__main__":
  main()
```

OUTPUT



16) Write a program to determine EOQ using various inventory models.

```
import math

def EOQ(D,A,H):
  Qstar = math.ceil(math.sqrt(2*D*A/H))
  tau = math.ceil(Qstar/D*365)
  aoc = D/Qstar*A
  ahc = Qstar/2*H
  tic = aoc+ahc
  print("EOQ :",Qstar)
  print("Average Inventory :",Qstar/2)
  print("Cycle time :",tau,"Days")
  print("Total inventory cost :",tic)

def EOQS(D,A,H,S):
  Qstar = math.ceil(math.sqrt(2*D*A*(H+S)/(H*S)))
  tau = math.ceil(Qstar/D*365)
  b= math.ceil(Qstar*(H/(H+S)))
  Td=math.ceil(((Qstar-b)/D)*365)
  Tb=math.ceil((b/D)*365)
  tsc=S*(b**2/(2*Qstar))
  tcc=H*((Qstar-b)**2/(2*Qstar))
  toc=A*(D/Qstar)
  tic=tsc+tcc+toc
  print("EOQ :",Qstar)
  print("Opt Backorder :",b)
  print("Time during which demand is met :",Td)
  print("Time during which demand is Backordered:",Tb)
  print("Cycle time :",tau,"Days")
  print("Total inventory cost :",tic)

def EPQS(D,A,H,P,S):
  d=1/365
  Qstar = math.ceil(math.sqrt((2*A*D*P*(H+S))/(H*S*(P-d))))
  print("EOQ :",Qstar)
  tc = math.sqrt((2*A*D*H*S*(P-d))/(P*(H+S)))
  print("Cost :",tc)

def EPQ(D,A,H,P):
  Qstar = math.ceil(math.sqrt((2*D*A)/(H*(1-(D/P)))))
  tau = math.ceil(Qstar/D*365)
  npr = D/Qstar
  apc = npr*A
  ahc = Qstar/2*(1-(D/P))*H
  tic=apc+ahc
  Imax = math.ceil(Qstar*(1-(D/P)))
  print("EPQ :",Qstar)
  print("Max Inventory :",Imax)
  print("Average Inventory :",math.ceil(Imax/2))
```

```
  print("Cycle time :",tau,"Days")
  print("Total inventory cost :",tic)

def main():
  D=float(input("Enter Demand : "))
  A=float(input("Enter Set-Up cost : "))
  H=float(input("Enter Holding cost : "))
  while(1):
    print("MENU")
    print("1. EOQ")
    print("2. EOQ (Shortage)")
    print("3. EPQ")
    print("4. FINITE PRODUCTION RATE AND SHORTAGES")
    print("5. CHANGE INPUTS")
    print("0. Exit")
    char = int(input("ENTER CHOICE : "))
    if(char==1):
      EOQ(D,A,H)
    elif(char==2):
      S=float(input("Enter Shortage cost : "))
      EOQS(D,A,H,S)
    elif(char==3):
      P=float(input("Enter the units produced : "))
      EPQ(D,A,H,P)
    elif(char==4):
      P=float(input("Enter the units produced : "))
      S=float(input("Enter Shortage cost : "))
      EPQS(D,A,H,P,S)
    elif(char==5):
      main()
    elif(char==0):
      exit()
    else :
      print("INVALID CHOICE")



if __name__=="__main__":
  main()
```

OUTPUT

C:\Windows\System32\cmd.exe

```
C:\Users\m\Documents\Python_MOR\Python Scripts>EOQ.py
Enter Demand : 10000
Enter Set-Up cost : 150
Enter Holding cost : .75
MENU
1. EOQ
2. EOQ (Shortage)
3. EPQ
4. FINITE PRODUCTION RATE AND SHORTAGES
5. CHANGE INPUTS
0. Exit
ENTER CHOICE : 1
EOQ : 2000
Average Inventory : 1000.0
Cycle time : 73 Days
Total inventory cost : 1500.0
MENU
1. EOQ
2. EOQ (Shortage)
3. EPQ
4. FINITE PRODUCTION RATE AND SHORTAGES
5. CHANGE INPUTS
0. Exit
ENTER CHOICE : 2
Enter Shortage cost : 2
EOQ : 2346
Opt Backorder : 640
Time during which demand is met : 63
Time during which demand is Backordered: 24
Cycle time : 86 Days
Total inventory cost : 1279.2043904518328
MENU
1. EOQ
2. EOQ (Shortage)
3. EPQ
4. FINITE PRODUCTION RATE AND SHORTAGES
5. CHANGE INPUTS
0. Exit
ENTER CHOICE : 5
Enter Demand : 800
Enter Set-Up cost : 10
Enter Holding cost : 4
```

17) Write a program to determine different characteristics using various queuing models.

```
import math
class Queue():
  def __init__(self):
    print("MENU")
    print("1. M/M/1 model")
    print("2. M/M/1/K model")
    print("3. M/M/C model")
    print("4. M/M/C/K model")
    self.choice=int(input("Enter choice : "))
    if(self.choice>4 or self.choice<1):
      print("Enter a valid choice ")
  def takeinput(self):
    self.l=float(input("Enter arrival rate (Lambda) : "))
    self.u=float(input("Enter departure rate (Mu) : "))
    self.p=self.l/self.u
  def model1(self):
    self.Ls=self.p/(1-self.p)
    self.Lq=self.p**2/(1-self.p)
    self.Ws=self.Ls/self.l
    self.Wq=self.Lq/self.l
  def model2(self,k):
    self.po=((1-self.p)/(1-self.p**(k+1)))
    self.pn=self.po*(self.p**k)
    if self.p==1:
      self.Ls=k/2
    else:
      self.Ls=((self.p/(1-self.p))-((k+1)(self.p(k+1))/(1-(self.p*(k+1)))))
      self.Lq=self.Ls-(self.l*(1-self.pn)/self.u)
      self.Ws=self.Ls/(self.l*(1-self.pn))
      self.Wq=self.Lq/(self.l*(1-self.pn))
  def model3(self,c):
    self.pod=0
    for i in range(1,c):
      self.pod=self.pod+((self.p**i)/math.factorial(i))
      self.pod=self.pod+((self.p*c)*c*self.u/(math.factorial(c)((c*self.u)-
self.l)))
      self.po=1/self.pod
      self.Lq=(((self.p*c)*self.l*self.u*self.po)/(math.factorial(c-1)((c*self.u-
self.l)**2)))
      self.Ls= self.Lq+self.p
      self.Wq=(((self.p*c)*self.u*self.po)/math.factorial(c-1)(((self.u*c)-
self.l))**2)
      self.Ws=self.Wq+(1/self.u)
  def model4(self,c,k):
    self.pod=0
    for i in range(1,c):
      self.pod=self.pod+((self.p**i)/math.factorial(i))
```

```
        self.pod=self.pod+((self.p*c)(k-c+1)/(math.factorial(c)))
        self.po=1/self.pod
        self.np=self.p/c
        self.ps=(self.p**c)*self.po/math.factorial(c)
        self.Lq=((((c*self.np)*self.np)*self.np*self.po(1-(self.np*(k-c+1))-((1-
self.np)(kc+1))(self.np(k-c))))/(math.factorial(c)((1-self.np)**2)))
        self.Ls=self.Lq+(self.p*(1-self.ps))
        self.Ws=self.Ls/(self.l*(1-self.ps))
        self.Wq=self.Ws-(1/self.u)
    def result(self):
      print("Ls = ",self.Ls)
      print("Lq = ",self.Lq)
      print("Ws = ", self.Ws)
      print("Wq = ",self.Wq)
ob=Queue()
if ob.choice==1:
    ob.takeinput()
    ob.model1()
    ob.result()
if ob.choice==2:
    ob.takeinput()
    k=int(input("Enter system capacity (K) : "))
    ob.model2(k)
    ob.result()
if ob.choice==3:
    ob.takeinput()
    c=int(input("Enter number of servers (C) : "))
    ob.model3(c)
    ob.result()
if ob.choice==4:
    ob.takeinput()
    k=int(input("Enter system capacity (K) : "))
    c=int(input("Enter number of servers (C) : "))
    ob.model4(c,k)
    ob.result()
```

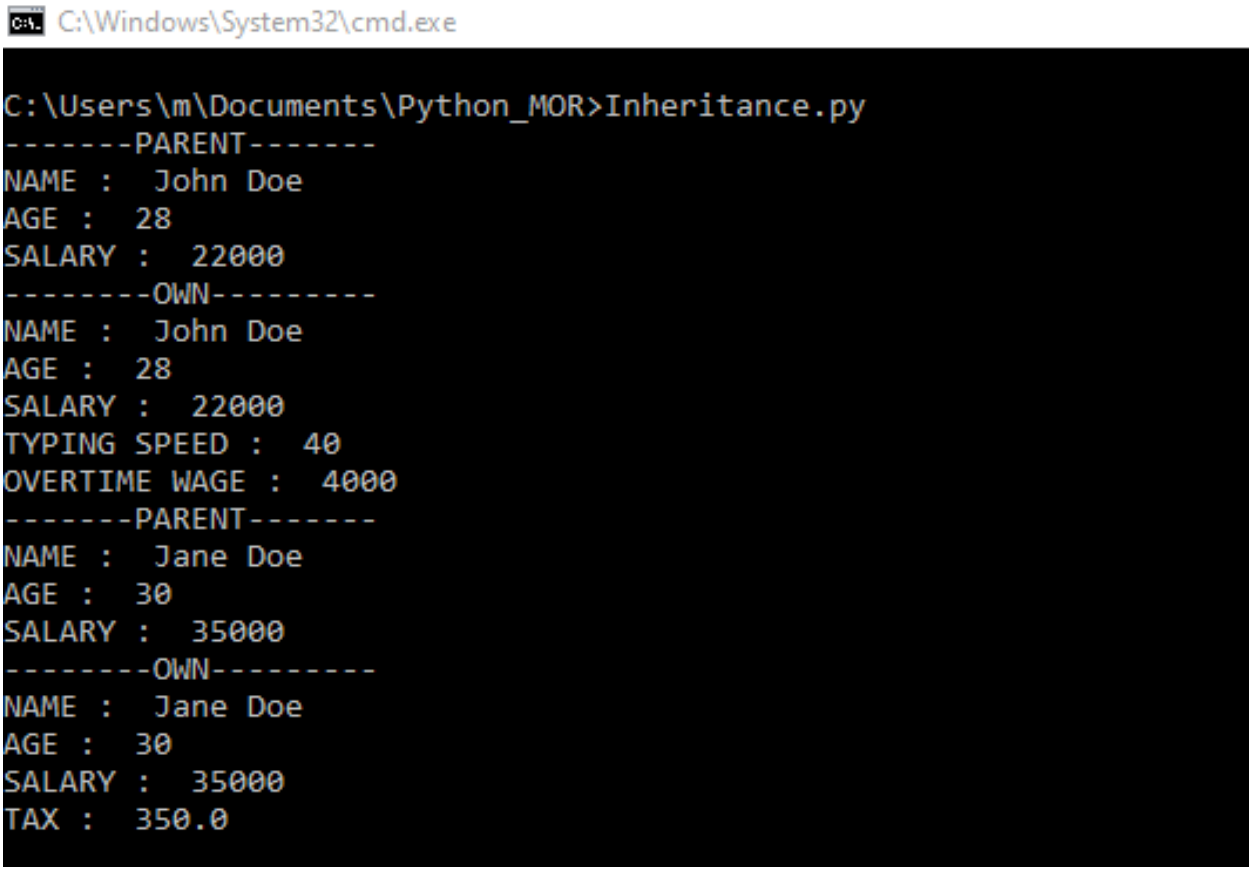
OUTPUT

```
C:\Windows\System32\cmd.exe

C:\Users\m\Documents\Python_MOR\Python Scripts>Queue.py
MENU
1. M/M/1 model
2. M/M/1/K model
3. M/M/C model
4. M/M/C/K model
Enter choice : 1
Enter arrival rate (Lambda) : 300
Enter departure rate (Mu) : 360
Ls =  5.000000000000002
Lq =  4.166666666666668
Ws =  0.016666666666666673
Wq =  0.013888888888888893
```

18) Write a program to implement Inheritance. Create a class Employee inherit two classes Manager and Clerk from Employee.

```
class Employee:
  def __init__(self, name, age, salary):
    self.name = name
    self.age = age
    self.salary=salary

  def display(self):
    print("NAME : ", self.name)
    print("AGE : ", self.age)
    print("SALARY : ", self.salary)

class Clerk(Employee):
  def __init__(self, name, age, salary,typingspeed):
    Employee.__init__(self, name, age, salary)
    self.typingspeed = typingspeed

  def overtime(self):
    Employee.display(self)
    print("TYPING SPEED : ", self.typingspeed)
    print("OVERTIME WAGE : ", self.typingspeed*100)


class Manager(Employee):
  def __init__(self, name, age, salary):
    Employee.__init__(self, name, age, salary)
    self.t=1

  def tax(self):
    Employee.display(self)
    print("TAX : ", self.salary*self.t/100)

c=Clerk('John Doe',28,22000,40)
m=Manager('Jane Doe',30,35000)
print('PARENT'.center(20,'-'))
c.display()
print('OWN'.center(20,'-'))
c.overtime()
print('PARENT'.center(20,'-'))
m.display()
print('OWN'.center(20,'-'))
m.tax()
```

OUTPUT


C:\Windows\System32\cmd.exe
C:\Users\m\Documents\Python_MOR>Inheritance.py
-------PARENT-------
NAME : John Doe
AGE : 28
SALARY : 22000
--------OWN---------
NAME : John Doe
AGE : 28
SALARY : 22000
TYPING SPEED : 40
OVERTIME WAGE : 4000
-------PARENT-------
NAME : Jane Doe
AGE : 30
SALARY : 35000
--------OWN---------
NAME : Jane Doe
AGE : 30
SALARY : 35000
TAX : 350.0

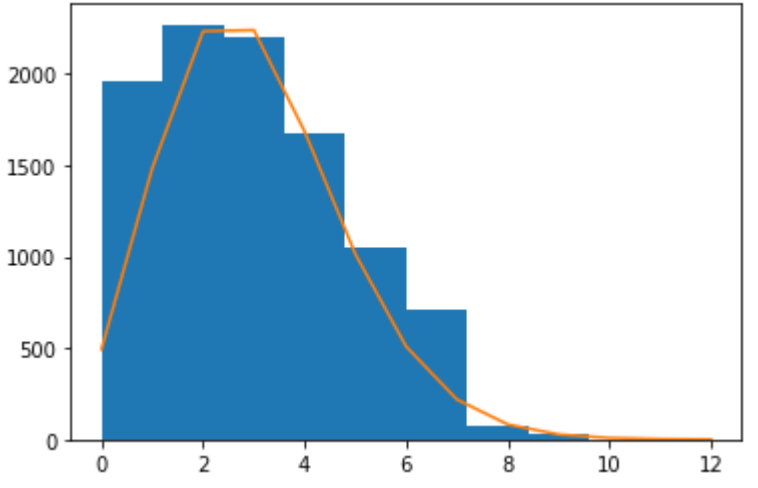
19) WAP to fit Poisson distribution on a given data.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
def poissonpmf(mu,x):
    return (np.exp(-mu) * (mu**x) / np.math.factorial(x))
```

In [3]:
```
param=3
size=10000
data=np.random.poisson(param, size)
mu=data.mean()
X = np.unique(data)
pi=np.array([poissonpmf(mu,xi) for xi in X])
plt.hist(data)
plt.plot(X,size*pi)
```

Out[3]:
```
[<matplotlib.lines.Line2D at 0x178c23d6488>]
```



20) Write a program to implement linear regression using python.

GRADIENT DESCENT

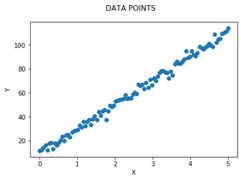
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
#N_ITERATIONS AND LEARNING RATE
iterations=1000
alpha=0.01
```

In [3]:
```
#DATA
m=100
x=np.linspace(0,5,m)
y=20*x+10+np.random.normal(0,2.5,np.size(x))
```

In [4]:
```
#DATA POINTS PLOT
fig = plt.figure()
plt.plot(x,y,'o')
fig.suptitle('DATA POINTS')
plt.xlabel('X')
plt.ylabel('Y')
```

Out[4]:
```
Text(0, 0.5, 'Y')
```



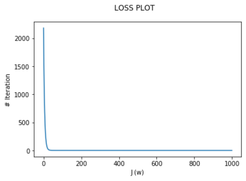
In [5]:
```
#INITIALIZING WEIGHTS
w0 = np.random.random()  #BIAS
w = np.random.rand(x[0].size)
```

In [6]:
```
#COST FUNCTION(LOSS)
def J(yh, y):
    return 0.5/m*np.sum((y-yh)**2)

#FUNCTION FOR UPDATING WEIGHTS
def updateW(w0,w,yh,y,Xi,alpha):
    dw0= -alpha*np.sum(yh-y)/m
    w0=w0+dw0
    dw = -alpha*np.sum(np.dot((yh - y), Xi))/m
    w = w+dw
    return w0,w

#ROOT MEAN SQUARE ERROR
def rmse(yh, y):
    return np.sqrt(((yh - y) ** 2).mean())
```

In [7]:
```
#FINDING WEIGHTS USING GD
costs = []
for i in range(iterations):
    yh=w0+w*x
    cost = J(yh,y)
    w0,w=updateW(w0,w,yh,y,x,alpha)
    costs.append(cost)
```

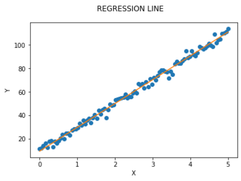
In [8]:
```
#LOSS PLOT
fig = plt.figure()
plt.plot(costs)
fig.suptitle('LOSS PLOT')
plt.xlabel('J (w)')
plt.ylabel('# Iteration')
```

Out[8]:
```
Text(0, 0.5, '# Iteration')
```



In [9]:
```
#REGRESSION LINE PLOT
fig = plt.figure()
plt.plot(x,y,'o',x,yh)
fig.suptitle('REGRESSION LINE')
plt.xlabel('X')
plt.ylabel('Y')
```

Out[9]:
```
Text(0, 0.5, 'Y')
```



In [10]:
```
#RMSE
rmseGD=rmse(yh,y)
rmseGD
```

Out[10]:
```
2.4994265555063975
```

LEAST SQUARES

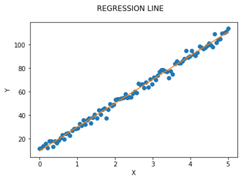
In [11]:
```
#CALC MEAN
x_mean=x.mean()
y_mean=y.mean()
```

In [12]:
```
#FIND WEIGHTS
sxy=(x*y).sum()
sxym=(x.sum()*y.sum())/m
sxx=(x*x).sum()
sxxm=(x.sum()*x.sum())/m
w=(sxy-sxym)/(sxx-sxxm)
w0=y_mean-w*x_mean
```

In [13]:
```
#YHAT
yh=w0+w*x
```

In [14]:
```
#PLOT
fig = plt.figure()
plt.plot(x,y,'o',x,yh)
fig.suptitle('REGRESSION LINE')
plt.xlabel('X')
plt.ylabel('Y')
```

Out[14]:
```
Text(0, 0.5, 'Y')
```



In [15]:
```
#RMSE
rmseLS=rmse(yh,y)
rmseLS
```

Out[15]:
```
2.495619883749647
```

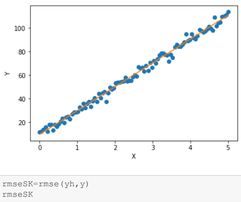
SKLEARN LINEAR REGRESSION

In [16]:
```
from sklearn.linear_model import LinearRegression
```

In [17]:
```
LR = LinearRegression()
x=x.reshape(-1,1)
LR.fit(x,y)
yh = LR.predict(x)
```

In [18]:
```
fig = plt.figure()
plt.plot(x,y,'o',x,yh)
fig.suptitle('REGRESSION LINE')
plt.xlabel('X')
plt.ylabel('Y')
```

Out[18]:
```
Text(0, 0.5, 'Y')
```



In [19]:
```
rmseSK=rmse(yh,y)
rmseSK
```

Out[19]:
```
2.495619883749647
```

COMPARE RMSE

In [20]:
```
print('RMSE GD = {}'.format(rmseGD))
print('RMSE LS = {}'.format(rmseLS))
print('RMSE SKLearn = {}'.format(rmseLS))
```

```
RMSE GD = 2.4994265555063975
RMSE LS = 2.495619883749647
RMSE SKLearn = 2.495619883749647
```

21) Write a program to perform read and write operation with .csv file

In [1]:
```
import csv
```

In [2]:
```
with open('NewData.csv', 'r') as file:
    reader=csv.reader(file)
    for row in reader:
        print(row)
```

```
['1', '2', '3', '4']
['1', '1', '22', '22']
['22', '19', '18', '14']
['49.895756', '17.775994', '5.27092', '0.771761']
['0.018632', '0.006864', '0.003923', '0.003923']
['0.486903', '0.100025', '1', '0']
['1', '1', '24', '24']
['22', '18', '16', '13']
['57.709936', '23.799994', '3.325423', '0.234185']
['0.003903', '0.003903', '0.003903', '0.003903']
['0.520908', '0.144414', '0', '0']
```

In [3]:
```
with open('NewData.csv', 'w', newline='') as file:
    writer=csv.writer(file)
    writer.writerow(['Attribute1', 'Attribute2', 'Class'])
    writer.writerow([11, 3, 0])
    writer.writerow([32, 3, 1])
```

USING PANDAS

In [4]:
```
import pandas as pd
```

In [5]:
```
df=pd.read_csv('NewData.csv')
df
```

Out[5]:

	Attribute1	Attribute2	Class
0	11	3	0
1	32	3	1

In [6]:
```
df.to_csv('Write.csv',index=False)
```

In [7]:
```
check=pd.read_csv('Write.csv')
check
```

Out[7]:

	Attribute1	Attribute2	Class
0	11	3	0
1	32	3	1

22) Write a Program to enter multiple values-based data in multiple columns/rows and show that data in python using DataFrames and pandas.

In [1]:
```
import pandas as pd
import random
```

In [2]:
```
print("Enter attribute 1")
att1 = [float(x) for x in input().split()]
print("Enter attribute 2")
att2 = [float(x) for x in input().split()]
print("Enter class")
cls = [float(x) for x in input().split()]
data = {'Attribute1' :att1,
        'Attribute2':att2,
        'Class':cls}
df = pd.DataFrame(data)
df
```

```
Enter attribute 1
1 2 3 4 5 6 7 8 9 10
Enter attribute 2
3 12 54 6 7 8 9 0 2 1
Enter class
0 1 0 0 0 1 0 0 1 0
```

Out[2]:

	Attribute1	Attribute2	Class
0	1.0	3.0	0.0
1	2.0	12.0	1.0
2	3.0	54.0	0.0
3	4.0	6.0	0.0
4	5.0	7.0	0.0
5	6.0	8.0	1.0
6	7.0	9.0	0.0
7	8.0	0.0	0.0
8	9.0	2.0	1.0
9	10.0	1.0	0.0

23) WAP in python to perform various statistical measures using pandas.

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df=pd.read_csv('RawData.csv')
df.head()
```

Out[2]:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	1	1	22	22	22	19	18	14	49.895756	17.775994	5.270920	0.771761	0.018632	0.006864	0.003923	0.003923	0.486903	0.100025
1	1	1	24	24	22	18	16	13	57.709936	23.799994	3.325423	0.234185	0.003903	0.003903	0.003903	0.003903	0.520908	0.144414
2	1	1	62	60	59	54	47	33	55.831441	27.993933	12.687485	4.852282	1.393889	0.373252	0.041817	0.007744	0.530904	0.128548
3	1	1	55	53	53	50	43	31	40.467228	18.445954	9.118901	3.079428	0.840261	0.272434	0.007653	0.001531	0.483284	0.114790
4	1	1	44	44	44	41	39	27	18.026254	8.570709	0.410381	0.000000	0.000000	0.000000	0.000000	0.000000	0.475935	0.123572

In [3]:
```
def statisticalmeasures(df):
    data=[]
    for i in df.columns:
        data.append(df[i].mean())
        data.append(df[i].median())
        data.append(df[i].mode()[0])
        data.append(df[i].count())
        data.append(df[i].std())
        data.append(df[i].max())
        data.append(df[i].min())
        data.append(df[i].quantile(0.75)-df[i].quantile(0.25))
    return data
```

In [4]:
```
data=np.array(statisticalmeasures(df))
StatsMeasure = pd.DataFrame(np.array_split(data, 20),
                            columns=['Mean','Median','Mode','Count','Std','Max','Min','IQR'],
                            index=[df.columns])
```

In [5]:
```
StatsMeasure
```

Out[5]:

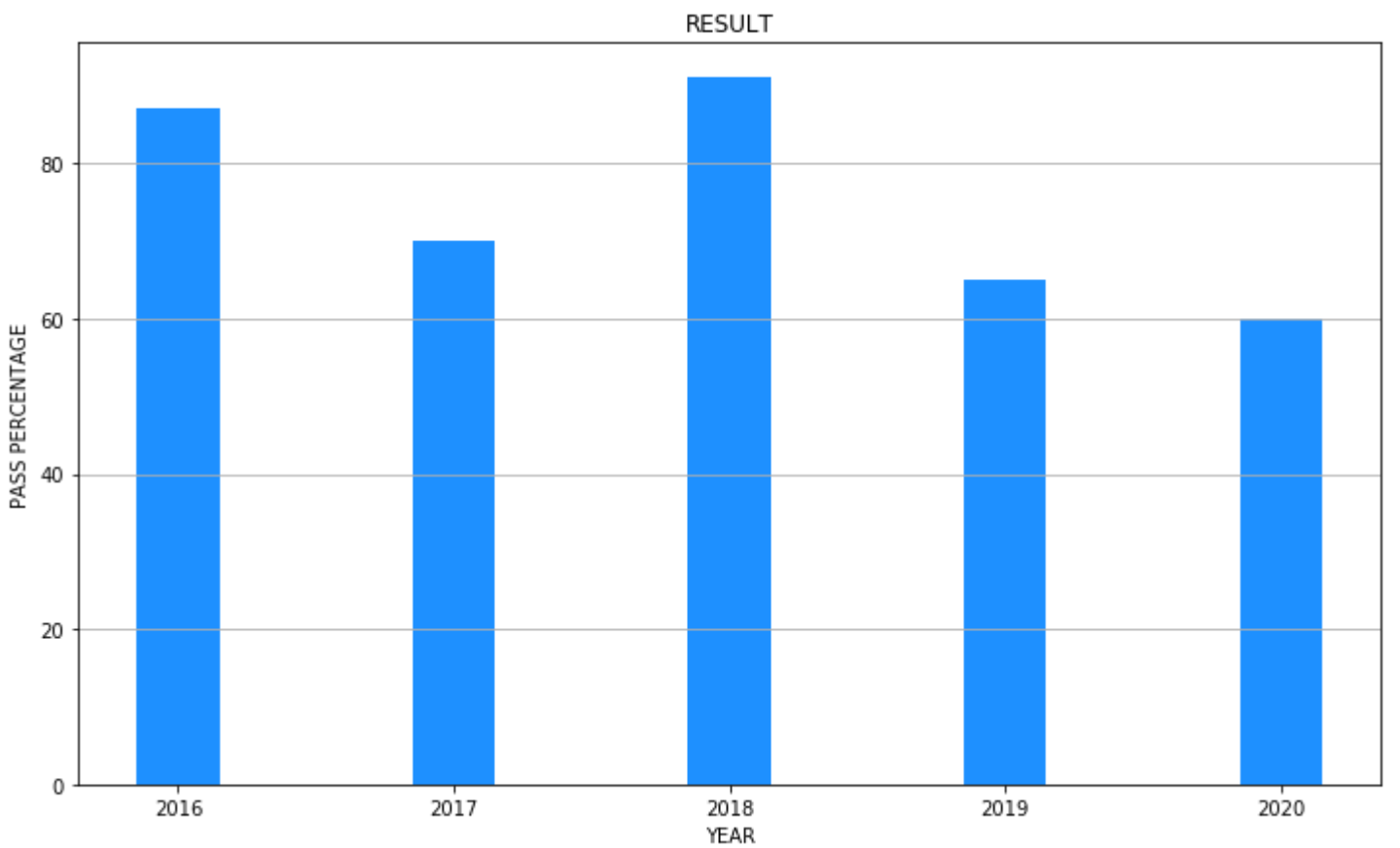
	Mean	Median	Mode	Count	Std	Max	Min	IQR
1	0.996230	1.000000	1.000000	1061.0	0.061314	1.000000	0.000000	0.000000
2	0.916117	1.000000	1.000000	1061.0	0.277343	1.000000	0.000000	0.000000
3	38.658812	36.000000	16.000000	1061.0	25.675524	151.000000	1.000000	40.000000
4	37.137606	35.000000	15.000000	1061.0	24.183066	132.000000	1.000000	38.000000
5	35.361923	32.000000	13.000000	1061.0	22.885503	120.000000	1.000000	37.000000
6	32.497644	30.000000	11.000000	1061.0	21.188935	105.000000	1.000000	35.000000
7	28.943450	25.000000	10.000000	1061.0	19.559696	97.000000	1.000000	33.000000
8	21.270500	18.000000	9.000000	1061.0	15.084584	89.000000	1.000000	24.000000
9	64.350017	45.003816	6.193941	1061.0	58.479527	403.939108	0.349274	65.334377
10	23.163337	17.293493	1.625616	1061.0	21.600927	167.131427	0.000000	23.437195
11	8.751308	4.563607	0.000000	1061.0	11.566846	106.070092	0.000000	10.549984
12	1.856494	0.513135	0.000000	1061.0	3.990004	59.766121	0.000000	1.829457
13	0.566788	0.023200	0.000000	1061.0	2.544188	51.423208	0.000000	0.195595
14	0.210432	0.001914	0.000000	1061.0	1.070305	20.098605	0.000000	0.041503
15	0.085072	0.000000	0.000000	1061.0	0.402848	5.937799	0.000000	0.004810
16	0.036054	0.000000	0.000000	1061.0	0.177188	3.086753	0.000000	0.003848
17	0.522824	0.523010	0.486570	1061.0	0.028021	0.592217	0.367762	0.040455
18	0.108430	0.106623	0.107603	1061.0	0.018137	0.219199	0.057906	0.023941
19	0.333648	0.000000	0.000000	1061.0	0.471738	1.000000	0.000000	1.000000
20	0.534402	1.000000	1.000000	1061.0	0.499050	1.000000	0.000000	1.000000

24) Write a program to plot a bar chart in python to display the result of a school for five consecutive years.

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import random
```

In [2]:
```
def barplot(label,value,width=0.3):
    fig = plt.figure(figsize=(12,7))
    plt.bar(label,value,color='dodgerblue',width=width)
    plt.xlabel('YEAR')
    plt.ylabel('PASS PERCENTAGE')
    plt.title('RESULT' )
    plt.grid(axis='y')
```

In [3]:
```
year = range(2016,2021)
result = np.array(random.sample(range(50, 100), 5))
barplot(year,result)
```

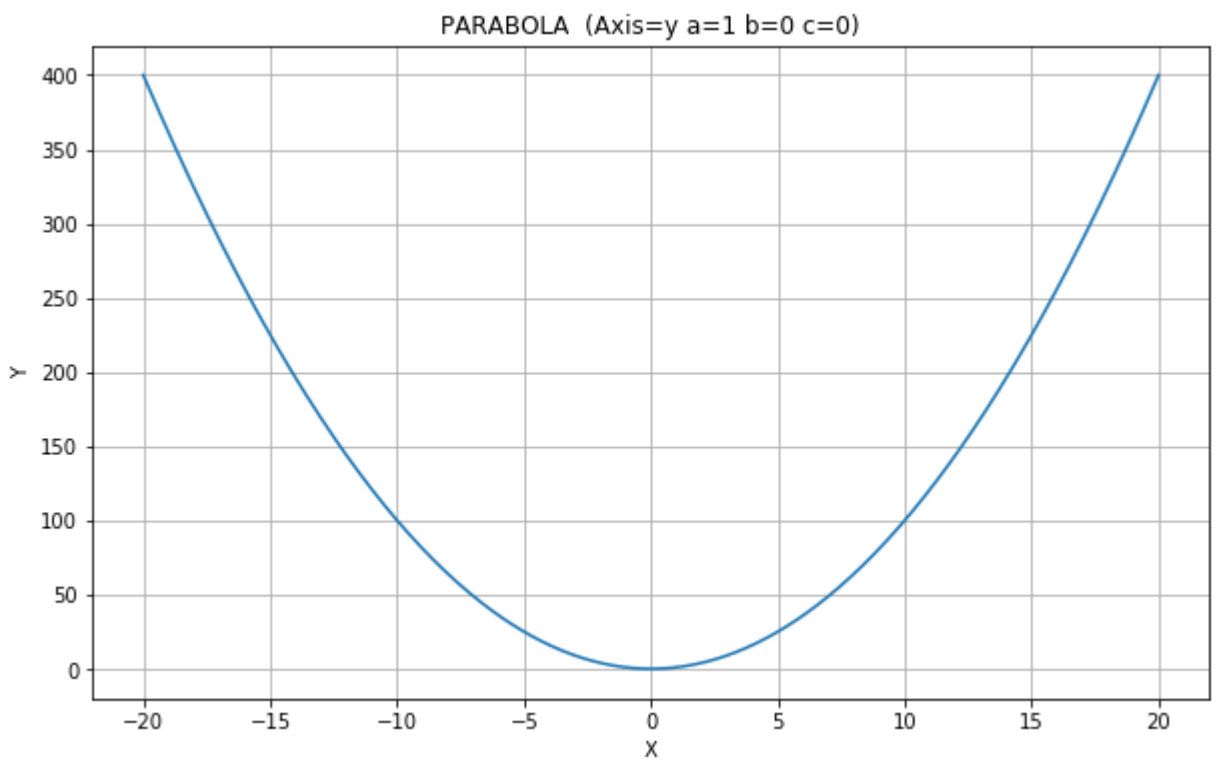


25) Write a program in python to plot a graph for the function y = x2

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
def parabola(min,max,a,b,c,ax_symmetry='y'):
    if (ax_symmetry=='y'):
        x=np.linspace(min, max, 500)
        y=a*(x**2)+b*x+c
    elif (ax_symmetry=='x'):
        y=np.linspace(min, max, 500)
        x=a*(y**2)+b*y+c
    else:
        print("WRONG INPUT")
        return
    fig = plt.figure(figsize=(10,6))
    plt.plot(x,y)
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.title('PARABOLA  (Axis={} a={} b={} c={})'.format(ax_symmetry,a,b,c))
    plt.grid(True)
```

In [3]:
```
parabola(-20,20,1,0,0,'y')
```



26) Write a program in python to plot a pie chart on consumption of water in daily life.

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import random
```

In [2]:
```
def pie(label,val,sa=90):
    fig = plt.figure(figsize=(16,9))
    plt.pie(val, labels = label ,startangle=sa, autopct=lambda x: '{:.2f}% ({:.0f})'.format(x,(x/100)*val.sum()))
    plt.title('Consumption of water in daily life')
```

In [3]:
```
purpose = ['Drinking','Cooking','Laundering','Cleaning','Miscellaneous']
usage = np.array(random.sample(range(10, 50), 5))
pie(purpose,usage)
```