**NUMBERS AND OPERATORS:**

a=-9

print(type(a))

b=4/2

print(type(b))

c=8+7j

print(type(c))

d=5.7

print(type(d))

x=5

y=23

print('x+y=',x+y)

print('x-y=',x-y)

print('x\*y=',x\*y)

print('x\*y=',x\*y)

print('x//y=',x//y)

print('x\*\*y=',x\*\*y)

print('x>y is',x>y)

print('x<y is',x<y)

print('x==y is',x==y)

print('x!=y is',x!=y)

print('x>=y is',x>=y)

print('x<=y is',x<=y)

x=True

y=False

print('x and y is',x and y)

print('x or y is',x or y)

print('not x is',not x)

x=13

y=10

print('x&y is',x&y)

print('x|y is',x|y)

print('~x is',~x)

print('x^y is',x^y)

print('x>>2 is',x>>2)

print('x<<2 is',x<<2)

x=3

y=x

print(y)

y+=x

print(y)

y-=x

print(y)

y\*=x

print(y)

y/=x

print(y)

y%=x

print(y)

y<<=x

print(y)

x=15

y=5

z=x

print(x is not y)

print(x is y)

x='Hai Python'

y={1:'a',2:'b'}

print('o' in x)

print('Python' not in x)

print(1 in y)

print('a' in y)

**STRING MANIPULATION:**

x='Python Program'

print x

letter=x[0]

print letter

len(x)

print(len(x))

print x.count('p')

print x.find('h')

print x.index('Program')

print x[0:5]

print x.count('')

print x.split('Program')

print x.startswith('P')

print x.endswith('m')

print "Program" \* 5

x.replace("Python","first")

string="python program"

print string.upper()

print string.lower()

print string.title()

print string.capitalize()

print string.swapcase()

print' '.join(reversed(string))

var1="python"

var2="program"

var3=var1+var2

print(var3)

**QUADRATIC EQUATION**

from math import sqrt

print("Quadratic function:a\*x^2+b\*x+c")

a=float(input("a:"))

b=float(input("b:"))

c=float(input("c:"))

r=b\*\*2-4\*a\*c

if r>0:

num\_roots=2

x1=(((-b)+sqrt(r))/(2\*a))

x2=(((-b)+sqrt(r))/(2\*a))

print("There are two roots:%f and %f"%(x1,x2))

elif r==0:

num\_roots=1

x=(-b)/2\*a

print("There is one root:",x)

else:

num\_roots=0

print("No roots,discriminat<0")

exit()

**BINARY SEARCH**

def binary\_search(arr,low,high,x):

if high>=low:

mid=(high+low)//2

if arr[mid]==x:

return mid

elif arr[mid]>x:

return binary\_search(arr,low,mid-1,x)

else:

return binary\_search(arr,mid+1,high,x)

else:

return -1

arr=[2,31,42,11,48]

x=4

result=binary\_search(arr,0,len(arr)-1,x)

if result!=-1:

print('Element is present at index',str(result))

else:

print('Element is not present in array')

def binary\_search(arr,x):

low=0

high=len(arr)-1

mid=0

while low<=high:

mid=(high+low)//2

if arr[mid]<x:

low=mid+1

elif arr[mid]>x:

high=mid-1

else:

return mid

return -1

arr=[9,4,7,15,42,68,33]

x=10

result=binary\_search(arr,x)

if result!=-1:

print('Element is present at index',str(result))

else:

print('Element is not present in array')

**LINEAR SEARCH:**

list1=[]

n=int(input('How many numbers to be added in the array:'))

print('Enter elements one by one')

for i in range(0,n):

list1.append(input())

print('\n\n created the linear search array')

print(list1)

def linear\_search(list1,n,key):

for i in range(0,n):

if(list1[i]==key):

return i

return -1

key=input('Element to be searched:')

print('Given Array is:',list1)

n=len(list1)

res=linear\_search(list1,n,key)

if(res==-1):

print('Element Not Found')

else:

print('Element Found At Index:',res)

**USER DEFINED FUNCTION:**

arr=[]

n=int(input('How many numbers to be added in the array:'))

print('Enter elements one by one')

for i in range(0,n):

arr.append(input())

print('\n\n created the Array')

print(arr)

def mergeSort(arr):

if len(arr)>1:

a=len(arr)//2

l=arr[:a]

r=arr[a:]

mergeSort(l)

mergeSort(r)

b=c=d=0

while b<len(l)and c<len(r):

if l[b]<r[c]:

arr[d]=l[b]

b+=1

else:

arr[d]=r[c]

c+=1

d+=1

while b<len(l):

arr[d]=l[b]

b+=1

d+=1

while c<len(r):

arr[d]=r[c]

c+=1

d+=1

def printList(arr):

for i in range(len(arr)):

print(arr[i])

print('Sorted Array')

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

mergeSort(arr)

print('Sorted Array is:')

printList(arr)

**LISTS:**

L=[]

n=int(input('How many numbers to be added in the list'))

print('Enter element one by one')

for i in range(0,n):

print('Enter number of index',i)

L.append(input())

print('\n\n Created List')

print(L)

m=int(input('Enter the position number of element to remove'))

L.pop(m)

print('\n\n list after removing selected elements')

print(L)

L.sort()

print('\n\n List in ascending order')

print(L)

L.reverse()

print('\n list in descending order')

print(L)

print('After sum()method')

print(sum(L))

print('Insert() method')

L.insert(5,92)

print(L)

print('After count()method')

print(L.count(66))

**ADDITION OF MATRIX**

rows=int(input("Enter the Numbers of rows:"))

column=int(input("Enter the Numbers of columns:"))

print("Enter the elements of first Matrix:")

matrix\_a=[[int(input()) for i in range(column)]

for i in range(rows)]

print("First Matrix is:")

for n in matrix\_a:

print(n)

print("Enter the elements of Second Matrix:")

matrix\_b=[[int(input())for i in range(column)]

for i in range(rows)]

print("Second Matrix is:")

for n in matrix\_b:

print(n)

result=[[0 for i in range(column)]

for i in range(rows)]

for i in range(rows):

for j in range(column):

result[i][j]=matrix\_a[i][j]+matrix\_b[i][j]

print("The sum of above two matrices is:")

for r in result:

print(r)

**SUBTRACTION OF TWO MATRIX**

rows=int(input("Enter the Numbers of rows:"))

column=int(input("Enter the Numbers of columns:"))

print("Enter the elements of first Matrix:")

matrix\_a=[[int(input()) for i in range(column)]

for i in range(rows)]

print("First Matrix is:")

for n in matrix\_a:

print(n)

print("Enter the elements of Second Matrix:")

matrix\_b=[[int(input())for i in range(column)]

for i in range(rows)]

print("Second Matrix is:")

for n in matrix\_b:

print(n)

result=[[0 for i in range(column)]

for i in range(rows)]

for i in range(rows):

for j in range(column):

result[i][j]=matrix\_a[i][j]-matrix\_b[i][j]

print("The difference of above two matrices is:")

for r in result:

print(r)

**MULTIPLICATION OF MATRIX**

rows\_a=int(input("Enter the Numbers of rows:"))

column\_a=int(input("Enter the Numbers of columns:"))

print("Enter the elements of first Matrix:")

matrix\_a=[[int(input()) for i in range(column\_a)]

for i in range(rows\_a)]

print("First Matrix is:")

for n in matrix\_a:

print(n)

column\_b=int(input("Enter the Numbers of columns for the second matrix:"))

print("Enter the elements of Second Matrix:")

matrix\_b=[[int(input())for i in range(column\_b)]

for i in range(column\_a)]

print("Second Matrix is:")

for n in matrix\_b:

print(n)

result=[[0 for i in range(column\_b)]

for i in range(rows\_a)]

for i in range(len(matrix\_a)):

for j in range(len(matrix\_b[0])):

for k in range(len(matrix\_b)):

result[i][j]+=matrix\_a[i][k]\*matrix\_b[k][j]

print("Matrix\_a X Matrix\_b is:")

for r in result:

print(r)

**PAY ROLL FOR EMPLOYEES USING TUPLES**

input\_tuple=final\_tuple=()

flag=False

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

print("Enter the name,empno and salary of",n,"employees")

for i in range(0,n):

name=raw\_input("Enter the name:")

id\_no=input("Enter the employee id:")

bpay=int(input("Enter the basic pay of employee:"))

da=float(bpay\*0.25)

hra=float(bpay\*0.15)

pf=float(bpay\*0.12)

ta=float(bpay\*0.075)

gpay=float(bpay+da+hra+ta)

netpay=float(gpay-pf)

input\_tuple=(name,id\_no,bpay,da,hra,pf,ta,gpay,netpay,)

final\_tuple=final\_tuple+(input\_tuple,)

print("\n")

print"id\_no\tname\tbpay\tda\thra\tpf\tta\tgpay\tnetpay"

for x in final\_tuple:

flag=True

print x[0],"\t",x[1],"\t",x[2],"\t",x[3],"\t",x[4],"\t",x[5],"\t",x[6],"\t",x[7],"\t",x[8]

if flag==False:

print("Record not found")

**MARKSHEET FOR STUDENTS USING TUPLES**

input\_tuple=final\_tuple=()

flag=False

n=int(input("Enter the Number of Students:"))

print("Enter the Name,Roll.No and Average Marks of 5 subjects of",n,"students")

for i in range(0,n):

name=raw\_input("Enter the Name")

roll\_no=input("Enter the Rollno:")

marks=input("Enter the Student Average Marks:")

per\_marks=(int(marks)\*100)/500

input\_tuple=(name,roll\_no,marks,per\_marks,)

final\_tuple=final\_tuple+(input\_tuple,)

input\_roll=input("Please enter the students roll number for results:")

for x in final\_tuple:

if x[1]==input\_roll:

flag=True

if int(x[3])in range(0,34):

print("Name:",x[0],"Roll\_Number:",x[1],"Failed")

elif int(x[3]) in range(35,49):

print("Name:",x[0],"Roll\_Number:",x[1],"Pass Class")

elif int(x[3]) in range(50,59):

print("Name:",x[0],"Roll\_Number:",x[1],"Second Class")

elif int(x[3]) in range(60,69):

print("Name:",x[0],"Roll\_Number:",x[1],"First Class")

elif int(x[3]) in range(70,100):

print("Name:",x[0],"Roll\_Number:",x[1],"First Class with Distinction")

if flag==False:

print("Record Not Found")

**REGULAR EXPRESSIONS USING VALIDATION**

import re

password=raw\_input("Enter the Password:")

while True:

if(len(password)<8):

flag=-1

break

elif not re.search("[a-z]",password):

flag=-1

break

elif not re.search("[A-Z]",password):

flag==-1

break

elif not re.search("[0-9]",password):

flag=-1

break

elif not re.search("[-@$]",password):

flag=-1

break

else:

flag=0

print("Valid Password")

break

if flag==-1:

print("Invalid Password")

**EXCEPTION HANDLING**

try:

num1=input("Enter the number 1:")

num2=input("Enter the number 2:")

add=num1+num2

sub=num1-num2

mul=num1\*num2

div=num1/num2

print(add)

print(sub)

print(mul)

print(div)

my\_list=[1,2,3]

i=int(input("Enter the index:"))

print(my\_list[i])

val=int(input("Enter positive number:"))

if val<0:

raise ArithmeticError

except TypeError:

print("Invalid,please provide number")

except ZeroDivisionError:

print("Denominator cannot be zero.. Please try again!!!")

except ArithmeticError:

print("This is Negative Number... Please Enter Positive Number!!")

else:

print("No Exception")

finally:

print("Execution Completed...")

**GUI Programming with Tkinter**

def exit():

window.destroy()

def convert():

c=int(e1.get())

f=((c\*9)/(5))+32

t1.config(state='normal')

t1.delete('1.0',tk.END)

t1.insert(tk.END,f)

t1.config(state='disabled')

import Tkinter as tk

window=tk.Tk()

window.geometry("300x250")

window.config(bg="#A569BD")

window.resizable(width=False,height=False)

window.title('Celsius TO Fahrenheit Converter!')

l1=tk.Label(window,text="Celsius To Fahrenheit Converter",font=("Arial",15),fg="White",bg="black")

l2=tk.Label(window,text="Enter temperature Celsius in :",font=("Arial",10,"bold"),fg="White",bg="#A569BD")

l3=tk.Label(window,text="Temperature in Fahrenheit is :",font=("Arial",10,"bold"),fg="White",bg="#A569BD")

empty\_l1=tk.Label(window,bg="#A569BD")

empty\_l2=tk.Label(window,bg="#A569BD")

e1=tk.Entry(window,font=("Arial",10))

btn1=tk.Button(window,text='Convert To Fahrenheit!',font=("Arial",10),command=convert)

btn2=tk.Button(window,text='Exit Application',font=("Arial",10),command=exit)

t1=tk.Text(window,state="disabled",width=15,height=0)

l1.pack()

l2.pack()

e1.pack()

empty\_l1.pack()

btn1.pack()

l3.pack()

t1.pack()

empty\_l2.pack()

btn2.pack()

window.mainloop()

**Network Programming:**

**Server.py**

import socket

import time

s=socket.socket()

host=socket.gethostname()

port=12309

s.bind((host,port))

s.listen(5)

while True:

c,addr=s.accept()

print 'Got Connection From', addr

c.send('Thank You For Connecting')

currentTime=time.ctime(time.time())+"\r \n"

c.send(currentTime.encode('ascii'))

c.close()

**client.py**

import socket

s=socket.socket()

host=socket.gethostname()

port=12309

s.connect((host,port))

print s.recv(1024)

tm=s.recv(1024)

s.close()

print("The time got from the server is %s"%tm.decode('ascii'))

**Database Connectivity**

**Create a table:**

import sqlite3

conn=sqlite3.connect('org.db')

print("Opened Database Successfully")

conn.execute('''CREATE TABLE ghostriderIn

(Id int primary key not null,

Name Text not null,

Age int not null,

Address char(50),

salary real)''')

print("Table Created Successfully")

conn.close()

**Insert Operation:**

import sqlite3

conn=sqlite3.connect('org.db')

print("Opened Database Successfully")

conn.execute("INSERT INTO ghostriderIn (Id,Name,Age,Address,Salary)\

VALUES(98,'fazil',21,'121A C21,NAGAR',20000),(941,'harish',22,'21GH NAGAR',60000),(543,'jesi',20,'SOUTH CARSTREET ',17000)")

conn.commit()

print("Records Created Successfully")

conn.close()

**Select Operation:**

import sqlite3

conn=sqlite3.connect('org.db')

print("Opened Database Successfully")

cursor=conn.execute("SELECT Id,Name,Address,Salary from ghostriderIn")

for row in cursor:

print "ID=",row[0]

print "Name=",row[1]

print "Address=",row[2]

print "Salary=",row[3],"\n"

print("Operation done Successfully")

conn.close()

**Update Operation:**

import sqlite3

conn=sqlite3.connect('org.db')

print("Opened Database Successfully")

conn.execute("UPDATE ghostriderIn set Salary=17000.0 where Id=543")

conn.commit()

print("Total Number of rows updated:",conn.total\_changes)

cursor=conn.execute("SELECT Id,Name,Address,Salary from ghostriderIn")

for row in cursor:

print "ID=",row[0]

print "Name=",row[1]

print "Address=",row[2]

print "Salary=",row[3],"\n"

print("Operation done Successfully")

conn.close()

**Delete Operation:**

import sqlite3

conn=sqlite3.connect('org.db')

print("Opened Database Successfully")

conn.execute("DELETE FROM ghostriderIn WHERE Id=98")

conn.commit()

print("Total Number of rows deleted:",conn.total\_changes)

cursor=conn.execute("SELECT Id,Name,Address,Salary from ghostriderIn")

for row in cursor:

print "ID=",row[0]

print "Name=",row[1]

print "Address=",row[2]

print "Salary=",row[3],"\n"

print("Operation done Successfully")

conn.close()