1. **# WAP to input a year and check whether the year is leap year or not**

y=int(input("Enter the year: "))

if y%400==0:

print("The year is a Century Leap Year")

elif y%100!=0 and y%4==0:

print("Year is a Leap year")

else:

print("Year is not a leap year")

**Enter the year: 2000**

**The year is a Century Leap Year**

2. **#WAP to input 3 numbers and print the greatest number using nested if**

a=float(input("Enter the first number: "))

b=float(input("Enter the second number: "))

c=float(input("Enter the third number: "))

if a>=b:

if a>=b:

print("First number :",a,"is greatest")

if b>a:

if b>c:

print("Second number :",b,"is greatest")

if c>a:

if c>b:

print("Third number :",c,"is greatest")

**Enter the first number4**

**Enter the second number2**

**Enter the third number2.1**

**First number : 4.0 is greatest**

3.**# WAP to input value of x and n and print the series along with its sum**

x=float(input("Enter the value of x"))

n=float(input("Enter the value of n"))

i=1

s=0

while i<n:

y=x\*\*i

print(y, "+", end='')

s=s+y

i+=1

print(x\*\*n) #to print the last element of series

s=s+(x\*\*n) #to add the last element of series

print("Sum of series =", s)

**Enter the value of x4**

**Enter the value of n2**

**4.0 +16.0**

**Sum of series = 20.0**

4.**# WAP to input a number and check whether it is prime number or not**

n=int(input("Enter the number"))

c=1

for i in range(2,n):

if n%i==0:

c=0

if c==1:

print("Number is prime")

else:

print("Number is not prime")

**Enter the number29**

**Number is prime**

5.**#WAP to print fibonacci series upto n terms, also find sum of series**

n=int(input("Enter the number of terms in fibonacci series"))

a,b=0,1

s=a+b

print(a,b,end=" ")

for i in range(n-2):

print(a+b,end=" ")

a,b=b,a+b

s=s+b

print()

print("Sum of",n,"terms of series =",s)

**Enter the number of terms in fibonacci series10**

**0 1 1 2 3 5 8 13 21 34**

**Sum of 10 terms of series = 88**

6. **# WAP to print the patterns**

**#1 program to print pattern**

for i in range(1,6):

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

print(j,end="")

print()

**1**

**12**

**123**

**1234**

**12345**

**#2 program to print pattern**

for i in range(5,0,-1):

for j in range(i):

print('\*',end="")

print()

**\*\*\*\*\***

**\*\*\*\***

**\*\*\***

**\*\***

**\***

7.**#WAP to print a string and the number of vowels present in it**

st=input("Enter the string")

print("Entered string =",st)

st=st.lower()

c=0

v=['a','e','i','o','u']

for i in st:

if i in v:

c+=1

print("Number of vowels in entered string =",c)

**Enter the stringI am a good boi**

**Entered string = I am a good boi**

**Number of vowels in entered string = 7**

8.**#Write a program to read a file story.txt and print the contents of file along with number of vowels present in it**

f=open("story.txt",'r')

st=f.read()

print("Contents of file :")

print(st)

c=0

v=['a','e','i','o','u']

for i in st:

if i.lower() in v:

c=c+1

print("\*\*\*\*\*FILE END\*\*\*\*\*")

print()

print("Number of vowels in the file =",c)

f.close()

**Contents of file :**

**Python is an interpreted, high-level, general-purpose programming language.**

**Created by Guido van Rossum and first released in 1991.**

**Python's design philosophy emphasizes code readability.**

**Its language constructs and object-oriented approach aim to help programmers write clear, logical code.**

**\*\*\*\*\*FILE END\*\*\*\*\***

**Number of vowels in the file = 114**

9.**#Write a program to read a file book.txt print the contents of file along with numbers of words and frequency of word computer in it**

f=open("book.txt","r")

L=f.readlines()

c=c1=0

v=['a','e','i','o','u']

print("Contents of file :")

for i in L:

print(i)

j=i.split()

for k in j:

if k.lower()=="computer":

c1=c1+1

for x in k:

if x .lower() in v:

c+=1

print("\*\*\*\*\*FILE END\*\*\*\*\*")

print()

print("Number of vowels in the file =",c)

print("Number of times 'computer' in the file =",c1)

f.close()

**Contents of file :**

**Python is an interpreted, high-level, general-purpose computer programming language.**

**Created by Guido van Rossum and first released in 1991.**

**Python's design philosophy emphasizes code readability.**

**Its language constructs and object-oriented approach aim to help programmers write clear, logical code.**

**\*\*\*\*\*FILE END\*\*\*\*\***

**Number of vowels in the file = 92**

**Number of times 'computer' in the file = 1**

10. **#Write a program to read a file Top.txt and print the contents of file along with the number of lines starting with A**

f=open("Top.txt","r")

st=f.readlines()

c=0

print("Contents of file :")

for i in st:

print(i)

if i[0]=="A":

c+=1

print("\n\*\*\*\*\*FILE END\*\*\*\*\*")

print()

print("Number of lines starting with 'A' =",c)

**Contents of file :**

**Python is an interpreted, high-level, general-purpose programming language.**

**Created by Guido van Rossum and first released in 1991.**

**Python's design philosophy emphasizes code readability.**

**Its language constructs and object-oriented approach aim to help programmers write clear, logical code.**

**\*\*\*\*\*FILE END\*\*\*\*\***

**Number of lines starting with 'A' = 0**

11.**#WAP to input a list of number and search for a given number using linear search**

l=eval(input("Enter the list of numbers"))

x=int(input("Enter the number"))

for i in l:

if i==x:

print("ELement present")

break

else:

print("Element not found")

**Enter the list of numbers[5,3,4,2,1]**

**Enter the number3**

**ELement present**

12.**#WAP to input a list of integers and search for a given number using binary search**

def bsearch(L,n):

start=0

end=len(L)-1

while start<=end:

mid=(start+end)//2

if L[mid]==n:

return True

elif L[mid]<=n:

start=mid+1

else:

end=mid-1

else:

return False

L=eval(input("Enter the list of numbers"))

n=int(input("Enter the number to find"))

L.sort()

if bsearch(L,n):

print("Element found")

else:

print("Element not found")

**Enter the list of numbers[5,3,4,2,1]**

**Enter the number to find6**

**ELement not found**

13.**#Write a function DigitSum() that takes a number and returns its digit sum**

def DigitSum(n):

s=0

n=str(n)

for i in n:

s=s+int(i)

return s

n=int(input("Enter the number"))

print("Sum of digits =",DigitSum(n))

**Enter the number69**

**Sum of digits = 15**

14.**#Write a function Div3and5() that takes a 10 elements numeric tuple and return the sum of elements which are divisible by 3 and 5**

def Div3and5(t):

s=0

for i in t:

if i%3==0 and i%5==0:

s=s+i

return s

#main

l=[]

for i in range(10):

print("Enter the ",i+1,"th number of the tuple",end=" ",sep="")

e=int(input())

l.append(e)

t=tuple(l)

print("Entered tuple :",t)

print("Sum of numbers in tuple divisible by 3 and 5 =",Div3and5(t))

**Enter the 1th number of the tuple 3**

**Enter the 2th number of the tuple 2**

**Enter the 3th number of the tuple 5**

**Enter the 4th number of the tuple 10**

**Enter the 5th number of the tuple 15**

**Enter the 6th number of the tuple 20**

**Enter the 7th number of the tuple 30**

**Enter the 8th number of the tuple 2**

**Enter the 9th number of the tuple 67**

**Enter the 10th number of the tuple 50**

**Entered tuple : (3, 2, 5, 10, 15, 20, 30, 2, 67, 50)**

**Sum of numbers in tuple divisible by 3 and 5 = 45**

15.**#Write a recursive function ChkPrime() that checks a number for Prime**

def ChkPrime(n,i):

while i<n:

if n%i==0:

return False

else:

i+=1

ChkPrime(n,i)

return True

n=int(input("Enter the number"))

if ChkPrime(n,2):

print("Number is prime")

else:

print("Number ain't prime")

**Enter the number23**

**Number is prime**

16.**#Write a function to input a list and arrange the list in ascending order using Bubble sort**

l=eval(input("Enter the list to arrange"))

for i in range(len(l)-1):

for j in range(len(l)-1):

if l[j]>l[j+1]:

l[j],l[j+1]=l[j+1],l[j]

print("Arranged list :",l)

**Enter the list to arrange[5,4,2,3,0,1]**

**Arranged list : [0, 1, 2, 3, 4, 5]**

17.**#Write a menu based program to demonstrate operation on a stack**

def isEmpty(stk):

if len(stk)==0:

return True

else:

return False

def push(stk,n):

stk.append(n)

def pop(stk):

if isEmpty(stk):

print("UNDERFLOW CONDITION")

else:

print("Deleted element:",stk.pop())

def peek(stk):

return stk[-1]

def display(stk):

if isEmpty(stk):

print("No Element Present")

else:

for i in range(-1,-len(stk)-1,-1):

if i==-1:

print("TOP",stk[i])

else:

print(" ",stk[i])

#main

stk=[]

while True:

print(" Stack operations")

print(" 1.PUSH")

print(" 2.POP")

print(" 3.PEEK")

print(" 4.DISPLAY STACK")

print(" 5.EXIT")

ch=int(input(" Enter the choice"))

if ch==1:

n=input("Enter the element to PUSH")

push(stk,n)

print("Element pushed")

elif ch==2:

pop(stk)

elif ch==3:

if isEmpty(stk):

print("UNDERFLOW CONDITION")

else:

print(peek(stk))

elif ch==4:

display(stk)

elif ch==5:

break

else:

print("INVALID CHOICE ENTERED")

print("THANKS FOR USING MY SERVICES")

**Stack operations**

**1.PUSH**

**2.POP**

**3.PEEK**

**4.DISPLAY STACK**

**5.EXIT**

**Enter the choice1**

**Enter the element to PUSH98**

**Element pushed**

**Stack operations**

**1.PUSH**

**2.POP**

**3.PEEK**

**4.DISPLAY STACK**

**5.EXIT**

**Enter the choice1**

**Enter the element to PUSH76**

**Element pushed**

**Stack operations**

**1.PUSH**

**2.POP**

**3.PEEK**

**4.DISPLAY STACK**

**5.EXIT**

**Enter the choice1**

**Enter the element to PUSH89**

**Element pushed**

**Stack operations**

**1.PUSH**

**2.POP**

**3.PEEK**

**4.DISPLAY STACK**

**5.EXIT**

**Enter the choice4**

**TOP 89**

**76**

**98**

**Stack operations**

**1.PUSH**

**2.POP**

**3.PEEK**

**4.DISPLAY STACK**

**5.EXIT**

**Enter the choice2**

**Deleted element: 89**

**Stack operations**

**1.PUSH**

**2.POP**

**3.PEEK**

**4.DISPLAY STACK**

**5.EXIT**

**Enter the choice4**

**TOP 76**

**98**

18.**# Write a menu based program to demonstrate operations on queue**

def isEmpty(qu):

if len(qu)==0:

return True

else:

return False

def ENQUEUE(qu,item):

qu.append(item)

if len(qu)==1:

rear=front=0

else:

rear=len(qu)-1

front=0

def DEQUEUE(qu):

if isEmpty(qu):

print("UNDERFLOW CONDITION")

else:

a= qu.pop()

print("ELEMENT DELETED:",a)

def peek(stk):

return stk[-1]

def display(qu):

if isEmpty(qu):

print("NO ELEMENT PRESENT")

else:

for i in range(len(qu)):

if i==0:

print("FRONT",qu[i])

elif i==len(qu)-1:

print("REAR ",qu[i])

else:

print(" ",qu[i])

#main

qu=[]

while True:

print(" QUEUE OPERATIONS")

print(" 1.ENQUEUE")

print(" 2.DEQUEUE")

print(" 3.DISPLAY QUEUE")

print(" 4.PEEK")

print(" 5.EXIT")

ch=int(input(" Enter your choice"))

if ch==1:

x=input("Enter the element to be inserted")

ENQUEUE(qu,x)

print("ELEMENT HAS BEEN INSERTED")

elif ch==2:

DEQUEUE(qu)

elif ch==3:

display(qu)

elif ch==4:

if isEmpty(qu):

print("UNDERFLOW CONDITION")

else:

print(peek(qu))

elif ch==5:

break

else:

print("INVALID CHOICE ENTERED")

print("THANKS FOR USING MY SERVICES")

**QUEUE OPERATIONS**

**1.ENQUEUE**

**2.DEQUEUE**

**3.DISPLAY QUEUE**

**4.PEEK**

**5.EXIT**

**Enter your choice1**

**Enter the element to be insertedPUBG**

**ELEMENT HAS BEEN INSERTED**

**QUEUE OPERATIONS**

**1.ENQUEUE**

**2.DEQUEUE**

**3.DISPLAY QUEUE**

**4.PEEK**

**5.EXIT**

**Enter your choice1**

**Enter the element to be insertedCOC**

**ELEMENT HAS BEEN INSERTED**

**QUEUE OPERATIONS**

**1.ENQUEUE**

**2.DEQUEUE**

**3.DISPLAY QUEUE**

**4.PEEK**

**5.EXIT**

**Enter your choice1**

**Enter the element to be insertedNFS**

**ELEMENT HAS BEEN INSERTED**

**QUEUE OPERATIONS**

**1.ENQUEUE**

**2.DEQUEUE**

**3.DISPLAY QUEUE**

**4.PEEK**

**5.EXIT**

**Enter your choice3**

**FRONT PUBG**

**COC**

**REAR NFS**

**QUEUE OPERATIONS**

**1.ENQUEUE**

**2.DEQUEUE**

**3.DISPLAY QUEUE**

**4.PEEK**

**5.EXIT**

**Enter your choice2**

**ELEMENT DELETED: NFS**

**QUEUE OPERATIONS**

**1.ENQUEUE**

**2.DEQUEUE**

**3.DISPLAY QUEUE**

**4.PEEK**

**5.EXIT**

**Enter your choice3**

**FRONT PUBG**

**REAR COC**

20**#Python –MYSQL Connectivity**

import mysql.connector

def insert():

cur.execute("select column\_name from information\_schema.columns where table\_name='{}'".format(tb))

print("Columns available in the table")

data=cur.fetchall()

for i in range(len(data)):

print("\t",i+1,". ",data[i][0],sep="")

b=""

for i in data:

print("NOTE: Please enter string/varchar/date values (if any) in quotes")

print("Enter the",i[0],end=" ")

a=input()

b=b+a+","

st=b.rstrip(",")

cur.execute("Insert into {} values({})".format(tb,st))

mycon.commit()

print("Record successfully inserted")

cur.execute("Select \* from {}".format(tb))

data=cur.fetchall()

print()

print("Updated Table given below")

for i in data:

print(i)

def display():

print("To display all records enter ALL otherwise")

n=input("Enter the number of records to display").upper()

cur.execute("Select \* from {}".format(tb))

if n=="ALL":

print(cur.fetchall())

else:

print(cur.fetchmany(int(n)))

def search():

cur.execute("select column\_name from information\_schema.columns where table\_name='{}'".format(tb))

print("Columns available in the table")

data=cur.fetchall()

for i in range(len(data)):

print("\t",i+1,". ",data[i][0],sep="")

w=input("Enter the column name using which you want to find the record")

print("Enter the value of that",w,end=" ")

wi=input()

cur.execute("select \* from {} where {}='{}'".format(tb,w,wi))

print(cur.fetchall())

def modify():

cur.execute("select column\_name from information\_schema.columns where table\_name='{}'".format(tb))

print("Fields available in the table")

data=cur.fetchall()

for i in range(len(data)):

print("\t",i+1,". ",data[i][0],sep="")

ch=input("Enter the field name to modify")

cur.execute("select \* from {}".format(tb))

print("All records given below")

for i in cur:

print(i)

cur.execute("select column\_name from information\_schema.columns where table\_name='{}'".format(tb))

print("Columns available in the table")

data=cur.fetchall()

for i in range(len(data)):

print("\t",i+1,". ",data[i][0],sep="")

w=input("Enter the column name using which you want to find the record")

print("Enter the value of that",w,end=" ")

wi=input()

print("Enter the new value of",ch,end="")

nw=input()

cur.execute("update {} set {}='{}' where {}='{}'".format(tb,ch,nw,w,wi))

mycon.commit()

print("Record sucessfully modified")

cur.execute("Select \* from {}".format(tb))

data=cur.fetchall()

print()

print("Updated Table given below")

for i in data:

print(i)

def delete():

cur.execute("select column\_name from information\_schema.columns where table\_name='{}'".format(tb))

print("Columns available in the table")

data=cur.fetchall()

for i in range(len(data)):

print("\t",i+1,". ",data[i][0],sep="")

w=input("Enter the column name using which you want to find the record")

print("Enter the value of that",w,end=" ")

wi=input()

cur.execute("Delete from {} where {}='{}'".format(tb,w,wi))

mycon.commit()

print("Record successfully deleted")

cur.execute("Select \* from {}".format(tb))

data=cur.fetchall()

print()

print("Updated Table given below")

for i in data:

print(i)

def create():

print("To create a new table")

t=input("Enter the new table name")

n=int(input("Enter the number of columns in new table"))

c=""

for i in range(n):

a=input("Enter the",(i+1),"th column name <space> data type[size] <space> constraints")

c=c+a+","

c=c.rstrip(",")

cur.execute("Create table {} ({})".format(t,c))

print("Table successfully created")

return t

#main begins

try:

hst=input("Enter the host")

usr=input("Enter tne user")

pd=input("Enter the password")

db=input("Enter the database")

mycon=mysql.connector.connect(host=hst,user=usr,database=db,passwd=pd)

cur=mycon.cursor()

if mycon.is\_connected():

print("Successfully Connected to Database")

try:

tb=input("Enter the table name to work in this session")

print()

cur.execute("Select \* from {}".format(tb))

print("Currently activated table you working on :")

print(cur.fetchall())

except:

print("No such table present")

print("Creating a new table for you...")

tb=create()

except:

print(“Error encountered”)

print("Possible causes of error:")

print(">> Invalid username entered")

print(">> Invalid host entered")

print(">> Invalid password entered")

print(">> Invalid database entered")

else:

while True:

print()

print("What do you want to do with your table?")

print("\t\t1. Insert Record")

print("\t\t2. Display Record")

print("\t\t3. Search Record")

print("\t\t4. Modify Record")

print("\t\t5. Delete Recod")

print("\t\t6. Exit")

print()

ch=int(input("Enter the choice"))

if ch==1:

insert()

elif ch==2:

display()

elif ch==3:

search()

elif ch==4:

modify()

elif ch==5:

delete()

elif ch==6:

mycon.close()

break

else:

print("Invalid choice entered")

print("Thanks for using our services")

**Enter the host localhost**

**Enter tne user root**

**Enter the password \*\*\*\*\***

**Enter the database Rishabh**

**Successfully Connected to Database**

**Enter the table name to work in this session job**

**Currently activated table you working on :**

**[(101, 'President', 12345), (102, 'Vice President', 125000), (103, 'Administration Assistant', 80000), (104, 'Accounting Manager', 70000), (105, 'Accountant', 65000), (106, 'Sales Manager', 80000), (107, 'President', 12345), (109, 'Teacher', 5000)]**

**What do you want to do with your table?**

**1. Insert Record**

**2. Display Record**

**3. Search Record**

**4. Modify Record**

**5. Delete Recod**

**6. Exit**

**Enter the choice1**

**Columns available in the table**

**1. jobid**

**2. JOBTITLE**

**3. salary**

**NOTE: Please enter string/varchar/date values (if any) in quotes**

**Enter the jobid 110**

**NOTE: Please enter string/varchar/date values (if any) in quotes**

**Enter the JOBTITLE "Manager"**

**NOTE: Please enter string/varchar/date values (if any) in quotes**

**Enter the salary 500000**

**Record successfully inserted**

**Updated Table given below**

**(101, 'President', 12345)**

**(102, 'Vice President', 125000)**

**(103, 'Administration Assistant', 80000)**

**(104, 'Accounting Manager', 70000)**

**(105, 'Accountant', 65000)**

**(106, 'Sales Manager', 80000)**

**(107, 'President', 12345)**

**(109, 'Teacher', 5000)**

**(110, 'Manager', 500000)**

**What do you want to do with your table?**

**1. Insert Record**

**2. Display Record**

**3. Search Record**

**4. Modify Record**

**5. Delete Recod**

**6. Exit**

**Enter the choice2**

**To display all records enter ALL otherwise**

**Enter the number of records to display4**

**[(101, 'President', 12345), (102, 'Vice President', 125000), (103, 'Administration Assistant', 80000), (104, 'Accounting Manager', 70000)]**

**What do you want to do with your table?**

**1. Insert Record**

**2. Display Record**

**3. Search Record**

**4. Modify Record**

**5. Delete Recod**

**6. Exit**

**Enter the choice5**

**Columns available in the table**

**1. jobid**

**2. JOBTITLE**

**3. salary**

**Enter the column name using which you want to find the recordjobid**

**Enter the value of that jobid 110**

**Record successfully deleted**

**Updated Table given below**

**(101, 'President', 12345)**

**(102, 'Vice President', 125000)**

**(103, 'Administration Assistant', 80000)**

**(104, 'Accounting Manager', 70000)**

**(105, 'Accountant', 65000)**

**(106, 'Sales Manager', 80000)**

**(107, 'President', 12345)**

**(109, 'Teacher', 5000)**

**What do you want to do with your table?**

**1. Insert Record**

**2. Display Record**

**3. Search Record**

**4. Modify Record**

**5. Delete Recod**

**6. Exit**

**Enter the choice4**

**Fields available in the table**

**1. jobid**

**2. JOBTITLE**

**3. salary**

**Enter the field name to modifysalary**

**All records given below**

**(101, 'President', 12345)**

**(102, 'Vice President', 125000)**

**(103, 'Administration Assistant', 80000)**

**(104, 'Accounting Manager', 70000)**

**(105, 'Accountant', 65000)**

**(106, 'Sales Manager', 80000)**

**(107, 'President', 12345)**

**(109, 'Teacher', 5000)**

**Columns available in the table**

**1. jobid**

**2. JOBTITLE**

**3. salary**

**Enter the column name using which you want to find the recordjobtitle**

**Enter the value of that jobtitle Teacher**

**Enter the new value of salary750000**

**Record sucessfully modified**

**Updated Table given below**

**(101, 'President', 12345)**

**(102, 'Vice President', 125000)**

**(103, 'Administration Assistant', 80000)**

**(104, 'Accounting Manager', 70000)**

**(105, 'Accountant', 65000)**

**(106, 'Sales Manager', 80000)**

**(107, 'President', 12345)**

**(109, 'Teacher', 750000)**