

Ex 1: Python in Interactive and Script Mode

(1)

Date: 29/01/24

LET'S TRY IT

From the Python Shell, enter the following and observe the results.

>>> 1024 ???	>>> -1024 ???	>>> .1024 ???
>>> 1,024 ???	>>> 0.1024 ???	>>> 1,024.46 ???

LET'S TRY IT

From the Python Shell, enter the following and observe the results.

>>> print('Hello') ???	>>> print('Hello') ???	>>> print('Let's Go') ???
>>> print("Hello") ???	>>> print("Let's Go!") ???	>>> print("Let's go!") ???

```
>>> print('Hello World')
Hello World
>>> print('Hello World\n')
Hello World

>>> print('Hello\nWorld')
Hello
World
>>> print('Hello\n\nWorld')
File "<stdin>", line 1
    print('Hello\n\nWorld')
    ^
SyntaxError: unterminated string literal (detected at line 1)
>>> print('Hello\n\nWorld')
Hello

World
>>> print('Hello World\n\n')
Hello World

>>> print('\nHello World')

Hello World
>>> print(1,'\n',2,'\n',3)
1
2
3
>>> print('\n',1,'\n',2,'\n',3)

1
2
3
```

LET'S TRY IT

From the Python Shell, enter the following and observe the results.

<code>>>> print('Hello World')</code> ???	<code>>>> print('Hello\nWorld')</code> ???
<code>>>> print('Hello World\n')</code> ???	<code>>>> print('Hello\n\nWorld')</code> ???
<code>>>> print('Hello World\n\n')</code> ???	<code>>>> print(1, '\n', 2, '\n', 3)</code> ???
<code>>>> print('\nHello World')</code> ???	<code>>>> print('\n', 1, '\n', 2, '\n', 3)</code> ???

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> python3
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('hello world')
hello world
>>> print('hello world\n')
hello world

>>> print('\nhello world')

hello world
>>> print(1, '\n', 2, '\n', 3)
1
2
3
>>> print('\n', 1, '\n', 2, '\n', 3)

1
2
3
>>> 
```

LET'S TRY IT

From the Python Shell, enter the following and observe the results.

```
>>> spring2014SemCredits = 15      >>> spring2014-sem-credits = 15
???
```

```
>>> spring2014_sem_credits = 15    >>> 2014SpringSemesterCredits = 15
???
```

```
>>> spring2014SemCredits=15
>>> spring2014_sem_credits=15
>>> spring2014-sem-credits=15
File "<stdin>", line 1
  spring2014-sem-credits=15
  ^^^^^^^^^^^^^^^^^^^^^^^^^
SyntaxError: cannot assign to expression here. Maybe you meant '==' instead of '='?
>>> 2014SpringSemesterCredits=15
File "<stdin>", line 1
  2014SpringSemesterCredits=15
  ^
SyntaxError: invalid decimal literal
>>>
```

2. Execute the following in interactive mode:

```
width = 17
height = 12.0
delimiter = 'a'
```

Execute each of the following expressions and write the value of the expression and the type of the values obtained:

- *width*/2
- *width*/2.0
- *height*/3
- 1 + 2 * 5
- *delimiter* * 5

```
>>> width=17
>>> height=12.0
>>> delimiter='a'
>>> width/2
8.5
>>> width/2.0
8.5
>>> height/3
4.0
>>> 1+2*5
11
>>> delimiter*5
'aaaaa'
>>> 
```

3. Use the Python interpreter as a calculator to answer the following:
- The volume of a sphere with radius r is $\frac{4}{3}\pi r^3$. What is the volume of a sphere with radius 5?
 - Suppose the cover price of a book is \$24.95, but bookstores get a 40% discount. Shipping costs \$3 for the first copy and 75 cents for each additional copy. What is the total wholesale cost for 60 copies?

```
>>> totalcost=(60*24.95)+3+(0.75*60)
>>> actualcost= totalcost-(0.40*totalcost)
>>> print(actualcost)
927.0
>>> █
```

```
>>> r=5
>>> vol=4/3*(3.14*r*r*r)
>>> print(vol)
104.66666666666666
>>>
```

4. Given the unit price of a product and the quantity of the product sold, find the total sale, using Python in script mode.



The image shows a code editor window titled '4.py' with a menu bar containing 'Open', 'Save', and window control buttons. The editor contains the following Python code:

```
1 unitprice=float(input("Enter the price of each product:"))
2 quantity=int(input("Enter the quantity sold:"))
3 totalsales=unitprice*quantity
4 print("The total sales is",totalsales)
5
```

Below the editor is a terminal window showing the execution of the script:

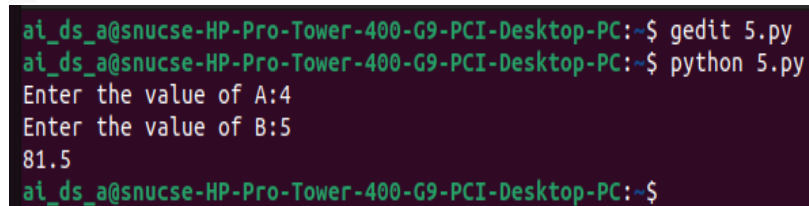
```
ai_ds_a@sncuse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~$ gedit 4.py
ai_ds_a@sncuse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~$ python3 4.py
Enter the price of each product:50
Enter the quantity sold:10
The total sales is 500.0
ai_ds_a@sncuse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~$
```

5. Write a program (script mode) to get two integers from a user, store them in variables a and b , and evaluate the following expression:

$$c = \frac{(a + b)^2 + 10}{a \times b}$$



```
Open 5.py Save
1 a=int(input("Enter the value of A:"))
2 b=int(input("Enter the value of B:"))
3 c=(a+b)**2 +10/(a*b)
4 print(c)
5
```



```
ai_ds_a@sncuse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~$ gedit 5.py
ai_ds_a@sncuse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~$ python 5.py
Enter the value of A:4
Enter the value of B:5
81.5
ai_ds_a@sncuse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~$
```

Ex. 2**EXPLORING OPERATORS AND EXPRESSIONS****Date: 29.01.24****Aim:**

To explore the use of operators and expressions in Python by writing programs for the following and executing them:

- a. Get the dimensions (floating point values) of a triangle, parallelogram, cylinder, cone, sphere, and rectangular prism, calculate each of their areas and display the result as a floating-point number approximated to 2 decimal places.
- b. Calculate the simple interest and compound interest, given the principal amount (P), rate of interest (R), term of deposit (T) (in years), and number of times interest is compounded in a year (n) from the user. Display the result rounded off to 4 decimal places.
- c. Calculate the salary of an employee of a company in terms of the basic pay, dearness allowance (DA), and house rent allowance (HRA). The DA and HRA are set as a certain percentage of the basic pay. Further, the company deducts 12% of the basic pay for PF. Compute the salary that would be received by an employee given the basic pay, percentage of basic pay for DA, and percentage of basic pay for HRA and print it, rounded off to the nearest integer. (Salary = Basic Pay + DA + HRA - PF)
- d. Implement the following using functions from the math module and print the results in the scientific notation, approximated to 2 decimal places. Get the values for the variables involved from the user:
 - (i) $A\cos\theta - B\sin(\theta)$
 - (ii) $A\cos(2\pi n)$
 - (iii) e^{an}
 - (iv) Euclidean distance between two points (x1, x2) and (y1, y2) (Use the formula for Euclidean distance - $\sqrt{(x1 - x2)^2 + (y1 - y2)^2}$)
 - (v) Convert an angle theta from radians to degrees.
 - (vi) Find the base 10 and base 2 logarithm of a floating-point number x.
- e. Obtain 2 decimal numbers from the user, display them in binary, octal, and hexadecimal forms, perform the logical operations (and, or, not, left and right shift) and print the results in binary and decimal forms.

Algorithm:**(a)**

Step 1: Get the input from the user

Step 2: Substitute in the necessary formula

Step 3: Display the output

(b)

Step 1: Get the input from the user

Step 2: Substitute in the formula

Step 3: Display the output

(c)

Step 1: Collect the salary details of the employee as input

Step 2: Calculate the total salary

Step 3: Display the output

(d)

Step 1: Get the input from the user

Step 2: Perform necessary operations

Step 3: Display the output

(e)

Step 1: Get the input from the user in decimal

Step 2: Convert to other forms using functions like `hex()`, `bin()`, `oct()`

Step 3: Display the output

(f)

Step 1: Get the values of a, b and c from the user

Step 2: Import `cmath` module and calculate discriminant

Step 3: Display the output

Program:

(a)

```
print("Enter the dimensions of TRIANGLE:")
b1=float(input("h:"))
h1=float(input("b:"))
a1=0.5*b1*h1
print("Area of triangle:",round(a1,2))
print("Enter the dimensions of PARALLELOGRAM:")
b2=float(input("b:"))
h2=float(input("h:"))
a2=b2*h2

print("Area of parallelogram:",round(a2,2))
print("Enter the dimensions of cylinder:") r3=float(input("r:"))
h3=float(input("h:")) a3=2*3.14*r3*(r3+h3)
print("Area of Cylinder:",round(a3,2)) print("Enter the dimensions of Cone:")
r4=float(input("r:"))
l4=float(input("l:")) a4=3.14*r4*(r4+l4)
```

```

print("Area of cone:",round(a4,2)) print("Enter the dimensions of sphere:")
r5=float(input("r:"))
a5=4*3.14*r5*r5
print("Area of sphere:",round(a5,2))
print("Enter the dimensions of rectangular prism:") w6=float(input("W:"))
l6=float(input("L:"))
h6=float(input("H:")) a6=2*(w6*l6+h6*l6+h6*w6)
print("Area of Rectangular prism:",round(a6,2))

```

(b)

```

p=int(input("Enter the principal amount:"))
r=int(input("Enter the Rate of interest:"))
t=int(input("Enter the term of deposit:"))
n=int(input("Enter the number of times interest is compounded in a year:"))
si=(p*r*t)/100
ci=p*(1+r/n)**(n*t)-p
print("SIMPLE INTEREST:",si)

print("COMPOUND INTEREST:",ci)

```

(c)

```

basic_pay=float(input("Enter the basic pay of the employee:"))
da_p=float(input("Enter the percentage of dearness allowance:"))
hra_p=float(input("Enter the percentage of house rent allowance:"))
pf=0.12*basic_pay
da=da_p/100*basic_pay
hra=hra_p/100*basic_pay
salary=basic_pay+da+hra-pf
print("Salary of the employee:",salary)

```

(d)

```

import math
a=int(input("Enter the value of A:"))
b=int(input("Enter the value of B:"))
pi=3.14
print('a')
theta=int(input("Enter the angle:"))
c=a*(math.cos(theta))-b*math.sin(theta)
print(c)
print('b')
n=int(input("Enter the value of n:"))
d=a*math.cos(2*pi*n)
print(d)
print('c')
print(math.exp(a*n))
print('d')
x1=int(input("Enter the x coordinate of point 1:"))

```



```

y1=int(input("Enter the y coordinate of point 1:"))

x2=int(input("Enter the x coordinate of point 2:"))
y2=int(input("Enter the y coordinate of point 2:"))
print("Euclidean Distance:",math.dist((x1,y1),(x2,y2)))
print('e')
theta1=int(input("Enter the angle in radians:"))
print("Angle in degrees:",math.degrees(theta1))
print('f')
x=int(input("Enter the value of x:"))
print(math.log(x,10))
print(math.log(x,10))

```

(e)

```

x1=int(input("Enter the first decimal number:"))
x2=int(input("Enter the second decimal number:"))
print("DISPLAYTING:")
print("Binary form:",bin(x1),bin(x2))
print("Octal form:",oct(x1),oct(x2))
print("Hexadecimal form:",hex(x1),hex(x2))
and_op= x1 and x2
or_op=x1 or x2
not_op1=not x1
not_op2=not x2
left=x1<<1
right=x2>>1
print("RESULT IN BINARY:")
print("AND:",bin(and_op))
print("OR:",bin(or_op))
print("NOT of X1:",bin(not_op1))
print("NOT of X2:",bin(not_op2))
print("LEFT SHIFT:",bin(left))

print("RIGHT SHIFT:",bin(right))
print("RESULT IN DECIMAL")
print("AND:",and_op)
print("OR:",or_op)
print("NOT of X1:",not_op1)
print("NOT of X2:",not_op2)
print("LEFT SHIFT:",left)
print("RIGHT SHIFT:",right)

```

(f)

```

import cmath
print("ax^2+bx+c")
a=int(input("Enter the value of a:"))
b=int(input("Enter the value of b:"))
c=int(input("Enter the value of c:"))
discr=b*b -4*a*c
dis=cmath.sqrt(discr)
if discr==0:
    print("It has a single solution:",-b/(2*a))

```

```
elif discri>0:
    print("It has 2 unique solutions:", -b+dis/(2*a), -b-dis/(2*a))
elif discri<0:
    print("It has conjugate roots")
print("First solution:", -b/(2*a), dis/(2*a))
print("second solution:", -b/(2*a), -dis/(2*a))
```

Screenshot of Output:(a)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit a.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 a.py
Enter the dimensions of TRIANGLE:
h:5
b:6
Area of triangle: 15.0
Enter the dimensions of PARALLELOGRAM:
b:2
h:5
Area of Parellelogram: 10.0
Enter the dimensions of cylinder:
r:1
h:3
Area of Cylinder: 25.12
Enter the dimensions of Cone:
r:6
l:5
Area of cone: 207.24
Enter the dimensions of sphere:
r:4
Area of sphere: 200.96
Enter the dimensions of rectangular prism:
W:1
L:2
H:5
Area of Rectangular prism: 34.0
```

(b)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 b.py
Enter the principal amount:100000
Enter the Rate of interest:2
Enter the term of deposit:5
Enter the number of times interest is compounded in a year:2
SIMPLE INTEREST: 10000.0
COMPOUND INTEREST: 102300000.0
```

(c)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit c.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 c.py
Enter the basic pay of the employee:250000
Enter the percentage of dearness allowance:5
Enter the percentage of house rent allowance:25
Salary of the employee: 295000.0
```

(d)

```

ai_ds_a@s nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python d.py
Enter the value of A:5
Enter the value of B:6
a
Enter the angle:60
-2.9332011754624814
b
Enter the value of n:2
4.999898538524866
c
22026.465794806718
d
Enter the x coordinate of point 1:1
Enter the y coordinate of point 1:2
Enter the x coordinate of point 2:3
Enter the y coordinate of point 2:4
Euclidean Distance: 2.8284271247461903
e
Enter the angle in radians:59
Angle in degrees: 3380.450991271857
f
Enter the value of x:2
0.30102999566398114
0.30102999566398114

```

(e)

```

ai_ds_a@s nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit e.py
ai_ds_a@s nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 e.py
Enter the first decimal number:526
Enter the second decimal number:364
DISPLAYTING:
Binary form: 0b10000001110 0b101101100
Octal form: 0o1016 0o554
Hexadecimal form: 0x20e 0x16c
RESULT IN BINARY:
AND: 0b101101100
OR: 0b10000001110
NOT of X1: 0b0
NOT of X2: 0b0
LEFT SHIFT: 0b100000011100
RIGHT SHIFT: 0b10110110
RESULT IN DECIMAL
AND: 364
OR: 526
NOT of X1: False
NOT of X2: False
LEFT SHIFT: 1052
RIGHT SHIFT: 182

```

(f)

```
ai_ds_a@s nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit f.py
ai_ds_a@s nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 f.py
ax^2+bx+c
Enter the value of a:1
Enter the value of b:1
Enter the value of c:2
It has conjugate roots
First solution: -0.5 1.3228756555322954j
second solution: -0.5 (-0-1.3228756555322954j)
```

Result:

Thus, programs have been written and executed to explore the use of operators and expressions in Python.

Ex. 3**EXPLORING CONDITIONAL STATEMENTS****Date: 05/02/24****Aim:**

To explore the use of conditional statements in Python by writing programs for the following and executing them:

- a. Create a simple calculator that accepts two numbers and an arithmetic operator as inputs and performs the appropriate operation on the given numbers and displays the result.
- b. Get a year from the user and check if it is a leap year and display the result.
- c. Find the maximum of three numbers, obtained from the user, using conditional statements.
- d. Obtain the marks secured by a student in Maths, Physics, Chemistry, Computer Science, and English, out of 100, and calculate their average. Check the range within which the average mark falls and display the appropriate grade. (A+ grade - 90 to 100, A grade - 80 to 90, B+ grade - 70 to 80, B grade - 60 to 70, C grade - 50 to 60, D grade - 40 to 50, F grade - less than 40)
- e. identify if a point (x,y) lies inside, outside, or on the circumference of a circle of radius "r", centered at the origin. Obtain the values of x, y, and r from the user.
- f. Obtain the lengths of the 4 sides of a quadrilateral and the angles at each corner of the quadrilateral. Verify if the dimensions represent a valid quadrilateral and if so, check whether the dimensions represent a square, a rectangle, or neither.

Algorithm:**(a)**

Step 1: Get the value of 2 integers from the user
Step 2: Get the mode of operation and perform it
Step 3: Compute the answer and display it

(b)

Step 1: Get the year from the user
Step 2: Check if it is a leap year using conditional statements
Step 3: Display the output

(c)

Step 1: Get 3 integers from the user
Step 2: Determine the largest value using conditional statements
Step 3: Display the output

(d)

Step 1: Get the marks from the user

Step 2: Find the average and check with conditions using elif statements

Step 3: Display the output

(e)

Step 1: Get the coordinates and radius from the user

Step 2: Compute the value of d

Step 3: Display the output

(f)

Step 1: Get the sides and angles of a quadrilateral from the user

Step 2: Check if it's a square or rectangle

Step 3: Display the output

Program:

(a)

```
a=int(input("Enter the first integer:"))
b=int(input("Enter the second integer:"))
op=input("Enter the operation to perform:(sum,difference,product,quotient)")

if(op=="sum"):
    print("Sum:",a+b)
elif(op=="difference"):
    print("Difference:",a-b)
elif(op=="product"):
    print("Product:",a*b)
elif(op=="quotient"):
    print("quotient:",a/b)
else:
    print("INVALID OPERATOR")
```

(b)

```
year=int(input("Enter the year:"))
if((year%4==0 and year%100!=0) or (year%400==0)):
    print("It\'s a leap year")
else:
    print("It\'s not a leap year")
```

(c)

```
a=int(input("Enter the value of a:"))
b=int(input("Enter the value of b:"))
c=int(input("Enter the value of c:"))
max=0
if a>=b:
    if a>c:
        max=a
    elif c>a:
        max=c
elif b>=a:
    if b>c:
        max=b
    elif c>b:
        max=c

print("The largest number is",max)
```

(d)

```
math=int(input("Enter marks scored in math:"))
phy=int(input("Enter marks scored in physics:"))
chem=int(input("Enter marks scored in chemistry:"))
cs=int(input("Enter marks scored in computer science:"))
eng=int(input("Enter marks scored in english:"))
avg=(math+phy+chem+cs+eng)/5
if 100>=avg>90:
    print("GRADE:A+")
elif 90>=avg>80:
    print("GRADE:A")
elif 80>=avg>70:
    print("GRADE:B")
elif 70>=avg>60:
    print("GRADE:B+")
elif 60>=avg>50:
    print("GRADE=C")
elif 50>=avg>=40:
    print("GRADE=D")
elif avg<40:
    print("GRADE:F")
```

(e)

```
import math
```



```

x=int(input("Enter the value of x coordinate:"))
y=int(input("Enter the value of y coordinate:"))
r=int(input("Enter the radius of the circle:"))
d=math.sqrt(x**2)+math.sqrt(y**2)
if d<r:
    print("The point lies inside the circle")
elif d>r:
    print("The point lies outside the circle")
elif d==r:
    print("The point lies on the circle")

```

(f)

```

print("Enter the dimensions of the quadrilateral:")
s1=int(input("Enter the length of side 1:"))
s2=int(input("Enter the length of side 2:"))
s3=int(input("Enter the length of side 3:"))
s4=int(input("Enter the length of side 4:"))
a1=int(input("Enter angle 1 in degrees:"))
a2=int(input("Enter angle 2 in degrees:"))
a3=int(input("Enter angle 3 in degrees:"))
a4=int(input("Enter angle 4 in degrees:"))
sum_angles=a1+a2+a3+a4
sum_sides=s1+s2+s3+s4
if sum_angles==360:
    print("It's a valid quadrilateral")
    if((s1==s2 and s2==s3 and s3==s4 and s1==s4) and (a1==a2 and a2==a3 and
a3==a4 and a4==90 and a1==a4)):
        print("It's a square")
    elif(((s1==s3 and s2==s4)or(s1==s2 and s3==s4)) and (a1==a2 and a2==a3 and
a3==a4 and a4==90 and a1==a4)):
        print("It's a rectangle")
else:
    print("It's not a quadrilateral")

```

Screenshot of Output:

(a)

```

PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/a.py"
Enter the first integer:5
Enter the second integer:6
Enter the operation to perform:(sum,difference,product,quotient)product
Product: 30

```

(b)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/b.py"
Enter the year:2023
It's not a leap year
```

(c)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/c.py"
Enter the value of a:10
Enter the value of b:2
Enter the value of c:5
The largest number is 10
```

(d)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/d.py"
Enter marks scored in math:90
Enter marks scored in physics:90
Enter marks scored in chemistry:90
Enter marks scored in computer science:90
Enter marks scored in english:90
GRADE:A
```

(e)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/e.py"
Enter the value of x coordinate:-3
Enter the value of y coordinate:2
Enter the radius of the circle:8
The point lies inside the circle
```

(f)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/f.py"
Enter the dimensions of the quadrilateral:
Enter the length of side 1:10
Enter the length of side 2:10
Enter the length of side 3:10
Enter the length of side 4:10
Enter angle 1 in degrees:90
Enter angle 2 in degrees:90
Enter angle 3 in degrees:90
Enter angle 4 in degrees:90
It's a valid quadrilateral
It's a square
```

Result:

Thus, programs have been written and executed to explore the use of conditional statements in Python.

Ex. 4**EXPLORING LOOPS****Date:**12/02/24**Aim:**

To explore the use of loops in Python by writing programs for the following and executing them:

- a. Print the list of prime numbers between 1 and N.
- b. Print the multiplication table up to M for a number N.
- c. Print the following pattern for 2N-1 rows.

```

      1
    2   2
  3   3   3
    2   2
      1

```

- d. Find the greatest common divisor of 2 numbers obtained from the user.
- e. Find the sum of the series $1, -(1^2), 2^3, -(3^4), 5^5, -(8^6), 13^7, \dots$ up to N terms.
- f. Find the sum of the digits of a given integer, N.
- g. Find the square root of an integer, N, using Newton's method. Obtain N and the limit, L, from the user.

Algorithm:

(a)

Step 1: Prompt the user to input an integer n .

Step 2: Initialize an empty list to store prime numbers.

Step 3: Loop through numbers from 2 to $n-1$:

- Initialize a counter for each number to track divisibility.
- Check divisibility of the number by iterating through possible divisors from 2 up to half the number.
- If the number is divisible by any divisor, increment the counter and break out of the loop.
- If the counter remains zero after checking all divisors, add the number to the list as a prime number.

Step 4: Print the list of prime numbers.

(b)

Step 1: Prompt the user to input integers n (number for which the table is to be printed) and m (number of multiples).

Step 2: Iterate from 1 to m :

- Print each multiplication of n with the current iterator i , displaying the result.

(C)

Step 1: Prompt the user for the number of rows n .

Step 2: Initialize a variable `space` equal to n to control leading spaces.

Step 3: For the upper half of the diamond (including the middle line):

- Print the necessary spaces to align the numbers centrally.
- Print numbers in ascending order from 1 up to the current row number, each repeated the same number of times as the row number.

- Decrease `space` by 1 after each row to reduce the number of spaces in the next line.

Step 4: Initialize another spacing variable `space1` for the lower half of the diamond.

Step 5: For the lower half of the diamond:

- Print the necessary spaces for alignment similar to the upper half.
- Print numbers in descending order from $n-1$ to 1, each number repeated as per its row number.
- Increase `space1` by 1 after each row to increase the number of spaces for alignment.

(d)

Step 1: Prompt the user to input two integers, a and b.

Step 2: Execute a while loop that continues as long as b is not zero:

- Inside the loop, set a to b and b to a modulo b, which repeatedly assigns b the remainder of a divided by b and a the old value of b.

Step 3: Once b becomes zero, print a as it now holds the GCD of the two numbers.

(e)

Step 1: Prompt the user to enter the number of terms, `n`, for the series.

Step 2: Initialize two variables `t1` and `t2` to represent the first two terms of the Fibonacci sequence, where `t1 = 0` and `t2 = 1`.

Step 3: Create an empty list `lst` and append `t2` to it to start the Fibonacci sequence.

Step 4: Generate the Fibonacci sequence up to `n` terms:

- Calculate the next term by adding `t1` and `t2`.
- Update `t1` to `t2` and `t2` to the new term (`c`).
- Append each new term to the list `lst`.

Step 5: Initialize `sum` to 0 and `num` to 0 to compute the series sum where each term is raised to an incrementally increasing power and alternately added or subtracted based on its index.

Step 6: Iterate over each Fibonacci number in `lst`:

- Use the index plus one as the exponent for (-1) to alternate the sign.
- Raise the Fibonacci number to the power of `num+1` (incrementing power).
- Accumulate these values in `sum`.

Step 7: After processing all terms, print the resulting `sum` of the series.

(f)

Step 1: Prompt the user to input an integer, `num`.

Step 2: Initialize `sum` to 0 to store the cumulative sum of the digits.

Step 3: Execute a while loop as long as `num` is not zero:

- Extract the last digit of `num` by taking $\text{num} \% 10$ and store it in `newnum`.
- Add `newnum` to `sum`.
- Remove the last digit from `num` by performing integer division $\text{num} // 10$.

Step 4: After exiting the loop, print the value of `sum`, which now contains the sum of all the digits of the input number.

(g)

Step 1: Define a function `newtons_method_sqrt` that takes two parameters: `N` (the number to find the square root of) and `L` (the tolerance level for approximation accuracy).

Step 2: Initialize `X` with the value of `N` to start the approximation.

Step 3: Enter a loop to repeatedly refine the approximation:

- Compute the next approximation `root` as the average of `X` and `N / X`.
- Check if the absolute difference between `root` and `X` is less than `L`. If it is, return `root` as the approximate square root.
- Update `X` to the new `root` for further refinement.

Step 4: Prompt the user to input the number `N` for which the square root is needed and the tolerance level `L`.

Step 5: Call `newtons_method_sqrt` with `N` and `L`, storing the result in `sqrt_approximation`.

Step 6: Print the approximate square root of `N`.

(a)

```
n=int(input("Enter n:"))
lst=[]

for i in range(2,n):
    count=0
    for x in range(2,int(i/2+1)):
        if i%x==0:
            count+=1
            break
    if count==0:
        lst.append(i)
print(lst)
```

(b)

```
n=int(input("Print the value of n:"))
m=int(input("Enter the value of m:"))
for i in range(1,n+1):
    print(n,"*", i, "=",n*i)
```

(c)

```
n=int(input("Enter the number of rows:"))
space=n
for i in range(n+1):
    for k in range(space):
        print(" ",end="")
    num=1
    for j in range(i):
```

```

        print(i,end=" ")
        num=num+1
    print()
    space-=1
space1=1
for i in range(n-1,0,-1):
    for k in range(space1):
        print(" ",end="")
    num=1
    for j in range(i):
        print(i,end=" ")
        num=num+1
    print()
    space1+=1

```

(D)

```

a=int(input("Enter the first integer:"))
b=int(input("Enter the second integer:"))
while b!=0:
    a,b=b,a%b
print("The GCD of two numbers is",a)

```

(e)

```

import math
sum=0
lst=[]
n=int(input("Enter the number of terms:"))
t1=0
t2=1
lst.append(t2)
for i in range(n):
    c=t1+t2
    t1=t2
    t2=c
    lst.append(c)
num=0
for i in lst:
    sum+=math.pow(-1,i+1)*math.pow(i,num+1)
    num+=1
print("The sum of the series is:",sum)

```

(f)

```

num=int(input("Enter the number"))
sum=0
newnum=0
while(num!=0):

```

```
newnum=int(num%10)
sum+=newnum
num=int(num/10)

print("The sum of the digits is:",sum)
```

(g)

```
def newtons_method_sqrt(N, L):
    X = N
    while True:
        root = 0.5 * (X + N / X)

        if abs(root - X) < L:
            return root

    X = root
N = int(input("Enter the value of N: "))
L = float(input("Enter the tolerance level L (<1): "))

sqrt_approximation = newtons_method_sqrt(N, L)
print(f"The square root is {N} is {sqrt_approximation}")
```


Screenshot of Output:

(a)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/a.py"
Enter n:12
[2, 3, 5, 7, 11]
```

(b)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara/lab_4$ gedit b.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara/lab_4$ python3 b.py
Print the value of n:10
Enter the value of m:10
10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
10 * 6 = 60
10 * 7 = 70
10 * 8 = 80
10 * 9 = 90
10 * 10 = 100
```

(c)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/c.py"
Enter the number of rows:4
  1
 2 2
3 3 3
4 4 4 4
 3 3 3
  2 2
   1
```

(d)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara/lab_4$ gedit d.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara/lab_4$ python3 d.py
Enter the first integer:24
Enter the second integer:36
The GCD of two numbers is 12
```

(e)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara/lab_4$ gedit e.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara/lab_4$ python3 e.py
Enter the number of terms:4
The sum of the series is: 3200.0
```

(f)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara/lab_4$ gedit f.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara/lab_4$ python3 f.py
Enter the number:1456
The sum of the digits is: 16
```

(g)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara/lab_4$ gedit g.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara/lab_4$ python3 g.py
Enter the value of N:36
Enter the tolerance level(<1):0.03
The square root is: 6.000000005333189
```

Result:

Thus, programs have been written and executed to explore the use of loops in Python.

Ex. 5**EXPLORING STRINGS****Date: 19/02/24****Aim:**

To explore strings in Python by writing programs for the following and executing them:

- a. Given a string, find the list of palindromic sub-strings without using built-in functions.
- b. List the number of sub-strings with the same first and last characters in a given string.
- c. Break a given string into multiple lines with N characters each. Get the value of N from the user.
- d. Sort the characters of a string based on their frequency of occurrence.
- e. Concatenate two strings of the same length, without using built-in functions, such that the characters of each string are placed alternately in the resultant string.
- f. Encode a given string using Caesar's cipher, where each letter in a string is replaced by a letter some fixed number of positions down the alphabet. Get the alphabetic string and the number of places each letter is to be shifted from the user.

Algorithm:

(a)

Step 1: Prompt the user to enter a string.

Step 2: Calculate the length of the string.

Step 3: Use nested loops to generate substrings:

- The outer loop determines the starting point of the substring.
- The inner loop determines the ending point of the substring.

Step 4: In each iteration of the inner loop, extract the substring from the current start to end index.

Step 5: Check if the first and last characters of the substring are the same.

Step 6: If they are the same, print the substring.

(b)

Step 1: Prompt the user to enter a string.

Step 2: Determine the length of the string.

Step 3: Identify the first character (a) and the last character (b) of the string.

Step 4: Use nested loops to generate and examine substrings:

- The outer loop sets the starting index of the substring.
- The inner loop sets the ending index of the substring.

Step 5: Extract each substring using the current start and end indices.

Step 6: Check if the first character of the substring matches a and the last character matches b.

Step 7: If both conditions are met, print the substring.

(c)

Step 1: Prompt the user for a string and the segment length n.

Step 2: Determine the length of the string.

Step 3: Initialize a counter to track characters printed.

Step 4: Iterate through the string:

- Print each character without moving to a new line.
- Increment the counter.
- If the counter equals n, print a newline and reset the counter.

(d)

Step 1: Prompt the user for a string.

Step 2: Initialize an empty dictionary to keep track of character frequencies.

Step 3: Iterate through each character in the string:

- If the character is already in the dictionary, increment its count.
- If not, add it to the dictionary with a count of 1.

Step 4: Sort the dictionary items by frequency in descending order.

Step 5: Create an empty list to store characters sorted by frequency.

Step 6: For each character-frequency pair in the sorted list, add the character to the new list, repeated by its frequency.

Step 7: Print the list of characters sorted by their frequency.

(e)

Step 1: Prompt the user for two strings.

Step 2: Calculate the lengths of both strings.

Step 3: Check if the lengths are equal:

- If they are, iterate through the strings:
- Print characters from both strings alternately without adding new lines.

- If they are not equal, display a message indicating invalid input.

(f)

Step 1: Prompt the user for a string and the number of shifts.

Step 2: Determine the length of the string.

Step 3: Iterate through each character in the string:

- If the character is not a space:
 - Calculate its new ASCII value after applying the shift.
 - Adjust the ASCII value to wrap around if it goes beyond 'z' or 'Z' (handling uppercase and lowercase separately).
 - Convert the adjusted ASCII value back to a character.
 - Print the character without moving to a new line.
- If the character is a space, print a space.

Program:

(a)

```
string=input("Enter a string:")
length=len(string)
for i in range(length):
    for j in range(i+1,length+1):
        k=string[i:j]
    if(k[0]==k[-1]):
        print(k)
```

(b)

```
string=input("Enter a string:")
length=len(string)
a=string[0]
b=string[length-1]
for i in range(length):
    for j in range(i+1,length+1):
        k=string[i:j]
    if(k[0]==a and k[-1]==b):
        print(k)
```

(c)

```
string=input("Enter a string:")
n=int(input("Enter the value of n:"))
length=len(string)
count=0
for i in range(length):
    print(string[i],end="")
    count+=1
    if(count==n):
        print()
        count=0
```

(d)

```
string = input("Enter a string: ")
frequency = {}

for char in string:
    if char in frequency: frequency[char] += 1
else:
    frequency[char] = 1

sorted_chars = sorted(frequency.items(), key=lambda x: x[1], reverse=True)

lst2 = []
for char, freq in sorted_chars:
    lst2.extend([char] * freq)

print(lst2)
```

(e)

```
string1=input("Enter the first string:")

string2=input("Enter the second string:")
length1=len(string1)
length2=len(string2)
if(length1==length2):
    for i in range(length1):
        print(string1[i],end="")
        print(string2[i],end="")
else:
    print("Invalid string")
```

(f)

```
string=input("Enter the string:")
shift=int(input("Enter the number of shifts:"))
length=len(string)
for i in range(length):
    if(string[i]!=" "):
        ascii=ord(string[i])+shift
        if(ascii>=122):
            ascii=ascii-26
        elif(ascii>=90 and ascii<97):
            ascii=ascii-26
        char=chr(ascii)
        print(char,end="")
        ascii=0
    else:
        print(" ",end="")
```

Screenshot of Output:

(a)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit a.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 a.py
Enter a string:awewa
a
awewa
w
wew
e
w
a
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$
```

(b)

(c)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit b.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 b.py
Enter a string:akshara
a
aksha
akshara
a
ara
a
```

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit c.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 c.py
Enter a string:akshara
Enter the value of n:3
aks
har
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$
```

(d)

(e)

```
aakksshhaarraai_ds_a@s nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 e.py
Enter the first string:akshara
Enter the second string:ajsad
Invalid string
ai_ds_a@s nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 e.py
Enter the first string:aksh
Enter the second string:amir
aakmsihrai_ds_a@s nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$
```

(f)

```
ai_ds_a@s nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit f.py
ai_ds_a@s nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python f.py
Enter the string:python program
Enter the number of shifts:5
udymts uwtlwfrai_ds_a@s nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$
```

Result:

Thus, programs have been written and executed to explore strings in Python.

Ex. 6**EXPLORING USER-DEFINED FUNCTIONS****Date:26/02/24****Aim:**

To explore user-defined functions in Python by writing programs for the following:

- a. Write a function that can generate Pythagorean triplets up to N. Obtain N from the user and pass it as an argument to your function.
- b. Write a function to generate the sum of the sine series, $\sin(x)$ and another to generate the sum of the cosine series $\cos(x)$ up to N terms. These functions must have two arguments, x and N, where N is an optional argument. Here x is the angle in radians. Get the angle in degrees from the user and convert to radians. Pass this angle to the functions. The computation of the sum of the sine series also requires the computation of the factorial of a number, so write a function to compute factorial as well.
- c. Write a menu-driven program to perform the following tasks. Write a separate function for each task.
 - Check if a given integer is an Armstrong number.
 - Check if a given integer is a Deficient number.
 - Check if a given integer is a Palindrome.
 - Exit the program.

Algorithm:**(a)**

Step 1: Prompt the user for the upper limit of the triplet values

Step 2: Define a function named triplets that takes an integer n as input

Step 3: Use nested loops with variables m and n to generate sides a, b, and c where $a = m^2 - n^2$, $b = 2mn$, $c = m^2 + n^2$

Step 4: Check if a, b, and c form a Pythagorean triplet and c is less than the input limit

Step 5: Print valid triplets

Step 6: Call the triplets function with the user's input value

(b)

Step 1: Import the math module for any necessary mathematical operations

Step 2: Prompt the user for an angle in degrees and the number of terms in the series

Step 3: Convert the angle from degrees to radians

Step 4: Define a recursive function fact to calculate the factorial of a number

Step 5: Define the function sine that computes the sine of an angle using its Taylor series expansion

- Initialize sum to 0
- Use a loop to add terms of the series up to n, adjusting the sign and factorial for each term
- Print the sum after each term is added

Step 6: Define the function cos that computes the cosine of an angle using its Taylor series expansion

- Similar to sine but adjust the power and factorial for cosine terms
- Print the sum after each term is added

Step 7: Call the sine and cos functions with the number of terms and the angle in radians

(c)

Step 1: Import the math module

Step 2: Define the function armstrong to determine if a number is an Armstrong number

Step 3: Define the function deficient to check if a number is deficient

Step 4: Define the function palindrome to verify if a number is a palindrome

Step 5: Display a menu with options to check for Armstrong number, deficient number, palindrome, or exit the program

Step 6: Continuously prompt the user for their choice until they decide to exit

- If choice is 1, prompt for a number and call the armstrong function
- If choice is 2, prompt for a number and call the deficient function
- If choice is 3, prompt for a number and call the palindrome function
- If choice is 4, exit the loop and terminate the program
- If the input is invalid, display an error message

Program:

(a)

```
num=int(input("Enter the value of n:"))

def triplets(n):
    a,b,c=0,0,0
    for m in range(1,num+1):
        for n in range(1,m):
            a=(m*m)-(n*n)
            b=(2*m*n)
            c=(m*m)+(n*n)
            if((a*a)+(b*b)==(c*c) and c<num):
                print(a,b,c)
            else:
                break
    triplets(num)
```

(b)

```
import math
deg=int(input("Enter the angle in degrees:"))
n=int(input("Enter the number of inputs:"))
x=deg*3.14/180

def fact(num):
    if(num==0):
        return 1
    else:
        return num*fact(num-1)

def sine(n,x):
```

```

sum=0
for i in range(n):
    sum+= pow(-1,i)*pow(x,2*i+1)/fact(2*i+1)
    print("The sum of the sine series is:",sum)

def cos(n,x):
    sum=0
    for i in range(n):
        sum+=pow(-1,i)*pow(x,2*i)/fact(2*i)
        print("The sum of the cosine series is:",sum)

sine(n,x)
cos(n,x)

```

(c)

```

import math

def armstrong(num):
    ognum=num
    sum=0
    n=0
    while(num!=0):
        num=num//10
        n+=1

    num=ognum
    while ognum!=0:
        number=ognum%10
        sum+=math.pow(number,n)
        ognum=ognum//10
        if(sum==num):
            print("It's an armstrong number")
        else:
            print("It's not an armstrong number")

def deficient(num):
    sum=0
    for i in range(1,(num//2)+1):
        if(num%i==0):
            sum+=i
        if(sum<num):
            print("It's a deficient number")
        else:
            print("It's not a deficient number")

def palindrome(num):
    string=str(num)
    k=string[::-1]

```

```

    if(k==string):
        print("It's a palindrome")
    else:
        print("It's not a palindrome")

while(1):
    print("(1) Armstrong number:")
    print("(2) Deficient number:")
    print("(3) Palindrom:")
    print("(4) EXit")
    choice=int(input("Enter a choice:"))

    if(choice==1):
        num=int(input("Enter a number:"))
        armstrong(num)
    elif(choice==2):
        num=int(input("Enter a number:"))
        deficient(num)
    elif(choice==3):
        num=int(input("Enter a number:"))
        palindrome(num)
    elif(choice==4):
        break
    else:
        print("Invalid choice")

```

Screenshot of Output:

(a)

```

15 8 17
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit a.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 a.py
Enter the value of n:14
3 4 5
8 6 10
5 12 13
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$

```

(b)

```
Enter the angle in degrees:45
Enter the number of inputs:3
The sum of the sine series is: 0.7068613175145052
The sum of the cosine series is: 0.7077097187760416
ai_ds_a@sncse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$
```

(c)

```
ai_ds_a@sncse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit c.py
ai_ds_a@sncse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 c.py
(1) Armstrong number:
(2) Deficient number:
(3) Palindrom:
(4) EXit
Enter a choice:1
Enter a number:371
It's an armstrong number
(1) Armstrong number:
(2) Deficient number:
(3) Palindrom:
(4) EXit
Enter a choice:1
Enter a number:754
It's not an armstrong number
(1) Armstrong number:
(2) Deficient number:
(3) Palindrom:
(4) EXit
Enter a choice:2
Enter a number:32
It's a deficient number
(1) Armstrong number:
(2) Deficient number:
(3) Palindrom:
(4) EXit
Enter a choice:2
Enter a number:56
It's not a deficient number
(1) Armstrong number:
(2) Deficient number:
(3) Palindrom:
(4) EXit
Enter a choice:3
Enter a number:121
It's a palindrome
(1) Armstrong number:
(2) Deficient number:
(3) Palindrom:
(4) EXit
Enter a choice:3
Enter a number:89
It's not a palindrome
(1) Armstrong number:
(2) Deficient number:
(3) Palindrom:
(4) EXit
Enter a choice:4
ai_ds_a@sncse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$
```

Result:

Thus, programs have been written and executed to explore user-defined functions in Python.

Ex. 7**EXPLORING RECURSION****Date: 04/03/25****Aim:**

To explore recursions in Python by writing recursive functions for the following:

- a. Find the factorial of a given integer.
- b. Find the value of a number, x , raised to the power of another number, y .
- c. Write a menu driven program to generate the following sequences up to N terms by writing a recursive function for each:
 - (i) Fibonacci series
 - (ii) Arithmetic progression, given the starting number and the common difference
 - (iii) Geometric progression, given the starting number and the common ratio
 - (iv) Harmonic progression, given the starting number and the common difference
 - (v) Triangular number series
- d. Display the binary equivalent of a decimal number.
- e. Find the largest digit in a given integer.
- f. Compute nCr .
- g. Display the sequence of steps to be followed to solve the Tower of Hanoi problem, given the number of disks, N .

Algorithm:**(a)**

STEP 1: Prompt the user to input a number.

STEP 2: Define the function ``fact`` to calculate the factorial:

- If the number is 0, return 1, as the factorial of 0 is 1.
- Otherwise, return the product of the number and the factorial of the number minus one

(recursive call).

STEP 3: Invoke the ``fact`` function with the user's input and print the result, which is the factorial of the given number.

(b)

STEP 1: Prompt the user to input a base number (``x``) and an exponent (``y``).

STEP 2: Define the function ``power`` to compute the power recursively:

- If the exponent ``y`` is 0, return 1, since any number to the power of 0 is 1.
- Otherwise, return the base number ``x`` multiplied by the result of the ``power`` function called with ``x`` and ``y-1``.

STEP 3: Call the ``power`` function with the user's inputs for ``x`` and ``y``, and print the result.

(c)

STEP 1: Display a menu with options for generating Fibonacci series, arithmetic progression

(AP), geometric progression (GP), harmonic progression (HP), triangular numbers, and an exit option.

STEP 2: Prompt the user to enter their choice.

STEP 3: Based on the user's choice:

- If choice is 1: Prompt for the number of terms and display the Fibonacci series up to that number.
- If choice is 2: Prompt for the first term, common difference, and number of terms, then display the arithmetic progression.
- If choice is 3: Prompt for the first term, common ratio, and number of terms, then display the geometric progression.
- If choice is 4: Prompt for the first term, common difference, and number of terms, then display the harmonic progression.
- If choice is 5: Prompt for `n` and display triangular numbers up to `n`.
- If choice is 6: Break the loop and exit the program.
- For invalid choices: Print an error message and show the menu again for a valid selection.

STEP 4: Continuously loop back to STEP 1 unless the user selects the option to exit (choice 6).

This allows multiple operations without restarting the program

(D)

STEP 1: Prompt the user to enter a decimal number.

STEP 2: Define the function `binary` to perform the conversion:

- Check if the number is 0. If so, return immediately since the base case is the last call for non-zero values.
- Compute the remainder of the number divided by 2 (this gives the current binary digit).
- Print the remainder (binary digit) without moving to a new line.
- Recursively call the `binary` function with the quotient of the number divided by 2 (i.e., `num // 2`) to process the next higher binary digit.

STEP 3: Invoke the `binary` function with the user's input number

(e)

STEP 1: Prompt the user to enter a decimal number.

STEP 2: Define the function `largest` to determine the largest digit:

- Check if the current number `num` is zero. If so, return the largest found digit (`lar`).
- Extract the last digit of the number using the modulus operator (`num % 10`).
- If the extracted digit is larger than the current largest (`lar`), update `lar`.
- Recursively call the `largest` function with the quotient of the number (`num // 10`) to process the next digit and the updated largest digit.

STEP 3: Invoke the `largest` function with the user's input number and an initial `lar` value of 0.

.(f)

STEP 1: Prompt the user to input the total number of items, n , and the number of items to choose, r .

STEP 2: Define the `fact` function to calculate the factorial of a given number recursively

STEP 3: Define the `comb` function to calculate the combination of n items taken r at a time

STEP 4: Call the `comb` function with the values of n and r provided by the user, and print the result.

(g)

Step 1: Ask the user for the number of disks, N .

Step 2: Define the function `tower_of_hanoi` with parameters for the number of disks n ,

Step 3: Execute `tower_of_hanoi` with N , 'A' as source, 'C' as destination, and 'B' as auxiliary.

Program:

(a)

```
num=int(input("Enter the number:"))
def fact(num):
    if num==0:
        return 1
    else:
        return num*fact(num-1)
print(fact(num))
```

(b)

```
x=int(input("Enter the number:"))
y=int(input("Enter the power:"))

def power(x,y):
    if(y==0):
        return 1
    else:
        return x*pow(x,y-1)
print(power(x,y))
```

(c)

```
def fibonacci_recursive(n, a=0, b=1):
    if n == 0:
        return
    else:
```

```

        print(a, end=' ')
        fibonacci_recursive(n-1, b, a+b)

def print_fibonacci_series(n):
    print("Fibonacci Series up to", n, "terms:")
    fibonacci_recursive(n)
def ap(an,d,n):
    if n==0:
        return 0
    else:
        print(an+d,end=" ")
        return ap(an+d,d,n-1)

def gp(a,r,n):
    if n==0:
        return 0
    else:
        print(a*r,end=" ")
        return gp(a,r*r,n-1)

def hp(an,d,n):

    if n==0:
        return 0
    else:
        print(1/(an+d),end=" ")
        return hp(an+d,d,n-1)

def triangular_number(n):
    if n == 1:
        return 1
    else:
        return n + triangular_number(n - 1)

while(1):
    print("(1) Fibonacci Series")
    print("(2) Arithmetic Progression")
    print("(3) Geometric Progression")
    print("(4) Harmonic Progression")
    print("(5) Triangular Number series")
    print("(6) Exit")

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

    if choice==1:
        n=int(input("Enter the number of inputs:"))
        print_fibonacci_series(n)

    elif choice==2:
        an=int(input("Enter the first term:"))
        d=int(input("Enter the common difference:"))

```

```

n=int(input("Enter the number of terms:"))
print(an,end=" ")
ap(an,d,n)

elif choice==3:
    a=int(input("Enter the first term:"))
    r=int(input("Enter the common ratio:"))
    n=int(input("Enter the number of terms:"))
    print(a,end=" ")
    gp(a,r,n)

elif choice==4:

    an=int(input("Enter the first term:"))
    d=int(input("Enter the common difference:"))
    n=int(input("Enter the number of terms:"))
    hp(an,d,n)

elif choice==5:
    n = int(input("Enter the value of n: "))
    for i in range(1, n + 1):
        print(triangular_number(i), end=' ')

elif choice==6:
    break

else:
    print("Enter a valid option")

```

```

(d)
num=int(input("Enter the decimal number:"))

def binary(num):
    sum=0
    if num==0:
        return 1
    else:
        sum=num%2
        print(sum,end="")
        return binary(num//2)

```

(e)

```
num=int(input("Enter the number:"))
lar=0
def largest(num,lar):
    if(num==0):
        return lar
    else:
        number=num%10
        if(number>lar):
            lar=number
        return largest(num//10,lar)
print(largest(num,lar))
```

(f)

```
n=int(input("Enter the value of n:"))
r=int(input("Enter the value of R:"))
sum=0
def fact(num):
    if num==0:
        return 1
    else:
        return num*fact(num-1)
def comb(n,r):
    if n==0:
        return 1
    else:
        return fact(n)/(fact(r)*fact(n-r))
print(comb(n,r))
```

(g)

```
def tower_of_hanoi(n, s, d,a):
    if n == 1:
        print(f"Move disk 1 from {s} to {d}")
        return
    tower_of_hanoi(n-1, s, a, d)
    print(f"Move disk {n} from {s} to {d}")
    tower_of_hanoi(n-1, a, d, s)

N = int(input("Enter the number of disks:"))
tower_of_hanoi(N, 'A', 'C', 'B')
```

Screenshot of Output:

(a)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit a.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 a.py
Enter the number:5
120
```

(b)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit b.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 b.py
Enter the number:5
Enter the power:2
25
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$
```

(c)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit c.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 c.py
(1) Fibonacci Series
(2) Arithmetic Progression
(3) Geometric Progression
(4) Harmonic Progression
(5) Triangular Number series
(6) Exit
Enter your choice:1
Enter the number of inputs:5
Fibonacci Series up to 5 terms:
0 1 1 2 3 (1) Fibonacci Series
(2) Arithmetic Progression
(3) Geometric Progression
(4) Harmonic Progression
(5) Triangular Number series
(6) Exit
Enter your choice:2
Enter the first term:2
Enter the common difference:2
Enter the number of terms:5
2 4 6 8 10 12 (1) Fibonacci Series
(2) Arithmetic Progression
(3) Geometric Progression
(4) Harmonic Progression
(5) Triangular Number series
(6) Exit
Enter your choice:3
Enter the first term:1
Enter the common ratio:2
Enter the number of terms:2
1 2 4 (1) Fibonacci Series
(2) Arithmetic Progression
(3) Geometric Progression
(4) Harmonic Progression
(5) Triangular Number series
(6) Exit
Enter your choice:5
Enter the value of n: 10
1 3 6 10 15 21 28 36 45 55 (1) Fibonacci Series
(2) Arithmetic Progression
(3) Geometric Progression
(4) Harmonic Progression
(5) Triangular Number series
(6) Exit
Enter your choice:6
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$
```

(d)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit d.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 d.py
Enter the decimal number:45
101101ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$
```

(e)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit e.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 e.py
Enter the number:4596
9
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$
```

(f)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit f.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 f.py
Enter the value of n:5
Enter the value of R:2
10.0
```

(g)

```
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ gedit g.py
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$ python3 g.py
Enter the number of disks:5
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
Move disk 4 from A to B
Move disk 1 from C to B
Move disk 2 from C to A
Move disk 1 from B to A
Move disk 3 from C to B
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 5 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
Move disk 3 from B to A
Move disk 1 from C to B
Move disk 2 from C to A
Move disk 1 from B to A
Move disk 4 from B to C
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/akshara$
```

Result:

Thus, programs have been written and executed to explore recursive functions in Python.

Ex. 8**EXPLORING LISTS AND TUPLES****Date: 28/03/24****Aim:**

To explore lists and tuples in Python by writing programs for the following:

- a. Sort a list of numbers obtained from the user using Bubble sort, without using built-in functions. Find the kth largest number in the list. Obtain k from the user.
- b. Sort a list of numbers obtained from the user using Insertion sort, without using built-in functions.
- c. Perform a linear search on a given list to find a given element and return its position if present in the list, without using built-in functions.
- d. Use the binary search algorithm to check if a given element is present in a given tuple, without using built-in functions, except the “sorted” function.
- e. Count the frequency of occurrence of each element of a tuple and display the elements and their count. Find the elements that appear only once in the tuple. (Write 2 programs, one that makes use of tuple methods and functions, and another that does not.)

Algorithm:**(a)**

STEP 1: Prompt the user to input a list of elements.

STEP 2: Request the user to enter the value of k to determine the k-th largest element.

STEP 3: Implement the bubble sort algorithm:

- Iterate through the list multiple times.
- In each iteration, compare adjacent elements and swap them if they are in the wrong order.

STEP 4: Print the sorted list once all elements are in the correct order.

STEP 5: Retrieve and print the k-th largest element using the sorted list's indexing.

(b)

STEP 1: Prompt the user to input a list of elements.

STEP 2: Iterate through the list starting from the second element.

STEP 3: For each element (key), compare it with the elements before it.

STEP 4: Shift all elements greater than the key one position ahead.

STEP 5: Insert the key into its correct position.

STEP 6: Print the sorted list after all elements have been processed.

(c)

STEP 1: Prompt the user to input a list of elements.

STEP 2: Request the user to enter the element they want to search for.

STEP 3: Initialize a flag variable to 0, indicating the element has not been found.

STEP 4: Iterate over the list:

- If the current element matches the search element, store its index, set the flag to 1, and break the loop.

STEP 5: If the flag is set to 1 after the loop, print that the element is found and its position.

STEP 6: If the flag remains 0, print that the element is not found

(d)

STEP 1: Prompt the user to input a list of elements and convert this list to a tuple.

STEP 2: Request the user to enter the number they wish to search for in the tuple.

STEP 3: Sort the tuple to ensure it is in ascending order, necessary for binary search.

STEP 4: Define the `search` function to implement the binary search:

- Use two pointers, `low` and `high`, to maintain the current search range.
- Calculate the middle index and compare the middle element with the target number.
- Adjust the `low` or `high` pointers based on the comparison to narrow the search range.
- Return `1` if the element is found.

STEP 5: Call the `search` function with initial values for `low` (0) and `high` (length of the tuple minus one).

STEP 6: Print the result based on the return value of the `search` function; print "Element is found" if the function returns `1`, otherwise print "Element is not found".

(e)

STEP 1: Prompt the user to input elements, which are then stored in a list and converted to a tuple `tup1`.

STEP 2: Initialize two empty lists, `lst2` for tracking unique elements and `lst3` for elements that appear only once.

STEP 3: Iterate over each element in `tup1`:

- If the element is not already in `lst2`, append it to `lst2` and use the `count` method of the tuple to print the number of occurrences of the element.

STEP 4: Iterate over elements in `lst2`:

- Use the `count` method on the original list `lst1` to check if any element appears exactly once.
- Append elements with exactly one occurrence to `lst3`.

STEP 5: Print `lst3`, which contains the list of elements that occur only once in the tuple.

Program:

(a)

```
#bubble sort
lst=eval(input("Enter the elements :"))
k=int(input("Enter the value of k:"))
for j in range(len(lst)):
    for i in range(len(lst)-1) :
        if lst[i]>lst[i+1]:
            lst[i],lst[i+1]=lst[i+1],lst[i]
print("the sorted list is",lst)
print("The",k,"largest element is ",lst[len(lst)-k])
```

(b)

```
#insertion sort
lst=eval(input("Enter the elements in the list"))
for i in range(1,len(lst)):
    key=lst[i]
    j=i-1
    while(j>=0 and lst[j]>key):
        lst[j+1]=lst[j]
        j-=1
    lst[j+1]=key
print("The sorted
```

©

```
lst=eval(input("Enter the elements in the list:"))
num=int(input("Enter the element to search:"))
flag=0
for i in range(len(lst)):
    if lst[i]==num:
        index=i
        flag=1
        break
if(flag==1):
    print("The element is found at",index,"position")
else:
    print("The element is not found")
```

(d)

```
def search(tup,low,high,num):

    while (low<=high):
        middle=(low+high)//2
        if tup[middle]==num:
            return 1
        elif( tup[middle]<num):
            low=middle+1
        elif (num<tup[middle]):
            high=middle-1

lst=eval(input("Enter the elements:"))
num=int(input("Enter the number to search:"))
low=0
high=len(lst)
lst.sort()
tup=tuple(lst)

if(search(tup,low,high,num)):
    print("Element is found")
else:
    print("Element is not found")
```

(e)

```
#using methods

lst1=eval(input("Enter the elements in the tuple:"))
tup1=tuple(lst1)
lst2=[]
lst3=[]

for element in tup1:
    if element not in lst2:
        lst2.append(element)
        print("count of",element,"is",tup1.count(element))

for i in lst2:
    if lst1.count(i)==1:
        lst3.append(i)
print("list of elements with only one occurence:",lst3)
```

```
#without methods
frequency={}
lst1=eval(input("Enter the elements in the list"))
unique=[]
for i in lst1:
    if i in frequency:
        frequency[i]+=1
    else:
        frequency[i]=1
for element,count in frequency.items():
    if count==1:
        unique.append(element)

for i ,j in frequency.items():
    print("count of",i,"is",j)
print("Numbers with unique occurence is",unique)
```

Screenshot of Output:

(a)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/a.py"
Enter the elements :[1,4,6,8,5]
Enter the value of k:2
the sorted list is [1, 4, 5, 6, 8]
The 2 largest element is 6
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> █
```

(b)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/b.py"
Enter the elements in the list:[1,4,6,8,4,9,76]
The sorted list is: [1, 4, 4, 6, 8, 9, 76]
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> █
```

(c)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/c.py"
Enter the elements in the list:[23,5,62,77,56]
Enter the element to search:62
The element is found at 2 position
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/c.py"
Enter the elements in the list:[23,5,62,77,56]
Enter the element to search:85
The element is not found
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> █
```

(d)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/d.py"
Enter the elements:[1,2,4,6,3,2,1]
Enter the number to search:3
Element is found
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> █
```

(e)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/e.py"
Enter the elements in the tuple:[1,2,4,6,3,2,1]
count of 1 is 2
count of 2 is 2
count of 4 is 1
count of 6 is 1
count of 3 is 1
list of elements with only one occurrence: [4, 6, 3]
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> █
```

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/e2.py"
Enter the elements in the list[1,2,4,6,3,2,1]
count of 1 is 2
count of 2 is 2
count of 4 is 1
count of 6 is 1
count of 3 is 1
Numbers with unique occurrence is [4, 6, 3]
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> █
```

Result:

Thus, programs have been written and executed to explore lists and tuples in Python.

Ex. 9**EXPLORING NESTED LISTS****Date:25/03/14**

/

Aim:

To explore nested lists in Python by writing programs for the following:

- a. Obtain 2 matrices from a user in the form of nested lists and perform matrix multiplication and addition, without using built in functions.
- b. Given the names and grades for each student in a class, store them in a nested list and print the name(s) of any student(s) having the second lowest grade.
- c. Check if a given list contains nested lists.
- d. Find the list in a list of lists whose sum of elements is the highest.

Algorithm:**(a)**

STEP 1: Prompt for matrices A and B.
STEP 2: Set up a zero matrix for the product if dimensions match.
STEP 3: Check dimension compatibility.
STEP 4: Perform matrix multiplication if possible.
STEP 5: Print error if dimensions don't match.
STEP 6: Display the resulting matrix.

(b)

STEP 1: Prompt for the number of students and initialize lists for students, grades, and lowest grades.
STEP 2: Collect names and grades of students, storing each as a sublist in a list.
STEP 3: Extract grades from the list and sort them.
STEP 4: Identify the second lowest grade in the sorted list.
STEP 5: Find all students with the second lowest grade and store their details.
STEP 6: Print the names and grades of students with the second lowest grade.

(c)

STEP 1: Prompt the user to input a list and store it.
STEP 2: Define a function to check for nested lists within the provided list.
STEP 3: Iterate through each item in the list; if any item is a list, return True.
STEP 4: If the function returns True, print that the list contains a nested list.
STEP 5: If no nested list is found, print that the list does not contain any nested lists.

(d)

STEP 1: Prompt the user to input a list of lists and store it.\STEP 2: Initialize a variable to keep track of the highest sum and an empty list to store the sublist with the highest sum.
STEP 3: Iterate through each sublist in the list.
STEP 4: Calculate the sum of each sublist.
STEP 5: Compare the sum of the current sublist with the highest recorded sum.
STEP 6: If the current sum is greater, update the highest sum and replace the sublist in the final list.
STEP 7: Print the sublist with the highest sum and the value of the highest sum.

Program:

(a)

```
a=eval(input("Enter the elements in matrix A:"))

b=eval(input("Enter the elements in matrix B:"))
prod = [[0 for _ in range(len(b[0]))] for _ in range(len(a))]
if(len(a)==len(b[0])):
    for i in range(len(a)):
        for j in range(len(b[0])):
            for k in range(len(b)):
                prod[i][j]+=a[i][k]*b[k][j]
else:
    print("Multipliacion is not possible")
    for p in prod:
        print(p)
```

(b)

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

lst=[]
index=[]
low=[]
for i in range(n):
    name=input("Enter the name:")
    grade=int(input("Enter the grade"))
    lst.append([name,grade])
for x in lst:
    index.append(x[-1])
index.sort()

lowest=index[1]

for x in lst:
    if lowest==x[-1]:
        low.append(x)
print("The second lowest grade=",low)
```

(c)

```
lst1=eval(input("Enter the list"))

def check(lst1):
    for x in lst1:
        if(isinstance(x,list)):
            return True
    return False

if check(lst1):
    print("It contains nested list")
else:
```

```
print("It doesn't")
```

(d)

```
lst=eval(input("Enter the elements in the list"))

final=[0]
highest=0
for x in lst:

    sum=0
    for i in x:
        sum+=i
    if sum>highest:
        highest=sum
        final.pop()
        final.append(x)
print(final)
print(highest)
```

Screenshot of Output:

(a)

```
ai_ds_a@snu-nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$ /bin/python3 /home/ai_ds_a/Desktop/akshara/a.py
Enter the elements in matrix A:[[1,2,3],[4,5,6]]
Enter the elements in matrix B:[[1,2],[3,4],[5,6]]
[22, 28]
[49, 64]
ai_ds_a@snu-nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$
```

(b)

```
ai_ds_a@snu-nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$ /bin/python3 /home/ai_ds_a/Desktop/akshara/b.py
Enter the number of students:4
Enter the name:a
Enter the grade:45
Enter the name:b
Enter the grade:89
Enter the name:c
Enter the grade:89
Enter the name:d
Enter the grade:98
The second lowest grade= [['b', 89], ['c', 89]]
ai_ds_a@snu-nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$
```

(c)

```
ai_ds_a@snu-nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$ /home/ai_ds_a/Desktop/.conda/bin/python /home/ai_ds_a/Desktop/akshara/c.py
Enter the list[[1,2],[3,4]]
It contains nested list
ai_ds_a@snu-nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$ /home/ai_ds_a/Desktop/.conda/bin/python /home/ai_ds_a/Desktop/akshara/c.py
Enter the list[1, 2, 3, 4]
It doesn't
ai_ds_a@snu-nucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$
```

(d)

```
Enter the elements in the list[[1,2,3],[4,5,6]]
[[4, 5, 6]]
15
● ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$ /home/ai_ds_a/Desktop/.conda/bin/python /home/ai_ds_a/Desktop/akshara/d.py
Enter the elements in the list[[1,2],[6,7,8],[4,5]]
[[6, 7, 8]]
21
○ ai_ds_a@snuce-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$
```

Result:

Thus, programs have been written and executed to explore nested lists in Python.

Date:08/04/24

EXPLORING DICTIONARIES

Aim:

To explore dictionaries in Python by writing programs for the following:

- Obtain a string from the user and create a dictionary that contains each character in the string mapped to the number of times the character appears in the given string. Write programs with and without the count method.
- Morse Code defines a standard encoding where each letter is mapped to a series of dots and dashes. Assume that the list of codes for each English alphabet is as follows:
[".", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", ".---", "-.-", ".-..", "--", ".-", "---", ".--.", "--.-", ".-.", "...", "-", "...-", "-.-", "-.-.", "-.-.", "-.."]
Create a dictionary that maps each alphabet to the corresponding Morse code. Obtain a list of words from the user and display the Morse code corresponding to each word. Store the codes in a dictionary. If the user inputs the same word multiple times, fetch the corresponding code from the word dictionary instead of computing the code again. Find out if multiple words have the same Morse code and display them.
- Obtain the marks secured by N students in 5 subjects namely, Math, English, Physics, Chemistry, and Computer Science from the user. The marks secured by each student may be received in the form of a list from the user and stored in a dictionary. Using the data obtained, create a nested dictionary in the following format, which contains the marks, average mark, and rank of each student:
{Student1: {Marks: [50, 45, 36, 48, 50], Average: 45.8, Rank: 2}, Student 2: {Marks: [43, 44, 48, 30, 37], Average: 40.4, Rank: 3}, Student3: {Marks: [50, 46, 50, 50, 50], Average: 49.2, Rank: 1}}
- Create a dictionary grouping a sequence of key-value pairs into a dictionary of lists.
- Read N numbers from a user and create a dictionary having two keys, EVEN and ODD which point to a list of even and odd numbers respectively.

Algorithm:

(A)

WITHOUT COUNT:

STEP 1: Prompt the user to enter a string and read the input.

STEP 2: Split the string into individual characters to create a list.

STEP 3: Initialize an empty dictionary to store the frequency of each character.

STEP 4: Iterate through the list of characters. For each character, if it is already in the frequency dictionary, increment its count by 1. If not, add it to the dictionary with an initial count of 1.

STEP 5: Print the frequency dictionary to show how many times each character appears in the string.

WITH COUNT:

STEP 1: Prompt the user to enter a string and store this input.

STEP 2: Initialize an empty dictionary ``frequency`` to store character frequencies and an empty string ``string2`` to keep track of unique characters encountered.

STEP 3: Iterate through each character in the input string. Add each unique character to ``string2`` if it hasn't been added already.

STEP 4: For each unique character in ``string2``, calculate its frequency in the original string using the ``count`` method and update the ``frequency`` dictionary with this data.

STEP 5: Print the ``frequency`` dictionary, which now contains the frequency of each unique character in the input string.

(B)

STEP 1: Define a list of Morse code symbols corresponding to each letter from 'a' to 'z'.

STEP 2: Create a dictionary, ``morse_dict``, mapping each lowercase English letter to its corresponding Morse code using a list comprehension.

STEP 3: Prompt the user to enter a list of words and store this list in the variable ``lst``.

STEP 4: Initialize an empty dictionary ``word_dict`` to store the Morse code translation for each word.

STEP 5: Iterate over each word in the list. For each word, initialize an empty string ``morse`` to build its Morse code representation.

STEP 6: For each letter in the word, check if the letter is already in ``word_dict``. If it is, set ``morse`` to the Morse code translation stored in ``word_dict``. If not, concatenate the Morse code for the letter from ``morse_dict`` to ``morse``.

STEP 7: Update ``word_dict`` with the word as the key and its Morse code translation as the value.

STEP 8: Print ``word_dict``, which now contains each word from the list mapped to its Morse code representation.

(C)

STEP 1: Prompt the user to enter the number of students and initialize necessary data structures: ``student_marks`` for storing individual student details, ``rank`` list for rankings, and ``avgl`` list for averages.

STEP 2: Loop through each student to input their marks in Maths, English, Chemistry, Computer Science, and Physics. Calculate the average mark and append to the ``avgl`` list.

STEP 3: Store each student's marks, calculated average, and an initially empty rank in the ``student_marks`` dictionary using a unique key for each student.

STEP 4: Sort the ``avgl`` list in descending order to prepare for ranking.

STEP 5: Assign ranks based on the sorted averages. Update the rank in `student_marks` for each student corresponding to their average.

STEP 6: Print the `student_marks` dictionary, showing details including marks, average, and rank for each student.

(d)

STEP 1: Define a list of tuples, where each tuple contains a color as a string and a corresponding integer.

STEP 2: Initialize an empty dictionary `dict1` to store the aggregated sums for each color.

STEP 3: Iterate over each tuple in the list. For each tuple, check if the color is already a key in the dictionary `dict1`.

STEP 4: If the color exists in the dictionary, increment its associated value by the integer from the tuple. If it does not exist, add the color to the dictionary with its associated value set to the integer from the tuple.

STEP 5: Print the dictionary `dict1`, which now contains the sum of numbers associated with each color.

(E)

STEP 1: Prompt the user to enter a list of numbers and store this list in the variable `n`.

STEP 2: Initialize a dictionary `dict1` with keys "ODD" and "EVEN", both set to an initial count of 0.

STEP 3: Iterate through each number in the list `n`.

STEP 4: For each number, check if it is even by using the modulus operation `num % 2 == 0`. If true, increment the count for "EVEN" in `dict1`. If false, increment the count for "ODD".

STEP 5: Print the dictionary `dict1`, which now contains the counts of odd and even numbers from the list.

Program:

```
(A)#without using count
frequency={}
string=input("Enter a string")
lst1=[word for word in string]
for i in lst1:
    if i in frequency:
        frequency[i]+=1
    else:
        frequency[i]=1

print(frequency)
```

USING COUNT

```
string=input("Enter a string:")
lst=[word for word in string]
frequency={}
string2=""
for i in string:
    if i not in string2:
        string2+=i

for i in string2:
    frequency.update({i:string.count(i)})
print(frequency)
```

(b)

```
morse_code=[".-","-...","-.-","-..",".",".-","--","...","..",".---","-.-",
".-.-","--","-.", "---","-.","-.-","-.","...","-", ".-","...-",".-","-.-",
".-","-.-","-.-"]
morse_dict={chr(i+97):code for (i,code)in enumerate(morse_code)}

lst=eval(input("Enter the list of words"))
word_dict={}

for i in lst:
    morse=""
    for j in i:
        if j in word_dict.keys():
            morse=word_dict[j]
        else:
            morse+=morse_dict[j]
    word_dict.update({i:morse})

print(word_dict)
```

(c)

```
n=int(input("Enter the number of students"))
student_marks={}
rank=[]
avg1=[]
dict1={}
avg=0
for i in range(n):
    mark=[]
    print("Enter the details of student",i+1)
    math=int(input("Maths"))
    english=int(input("English:"))
```

```

physics=int(input("Physics"))
chemistry=int(input("Chemistry"))
cs=int(input("Computer"))
mark.append([math,english,physics,chemistry,cs])
avg=(math+english+physics+chemistry+cs)/5
avg1.append(avg)
student_marks[f"student{i+1}"]={"Marks":mark,"Average":avg,"Rank":''}

avg1.sort(reverse=True)
for avg in avg1:
    rank=n
    for students,info in student_marks.items():
        rank-=1
        if info["Average"]==avg:
            info["Rank"]=rank+1

print(student_marks)

```

(d)

```

lst=[('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]
dict1={}
for colour,number in lst:
    if colour in dict1.keys():
        dict1[colour]+=number
    else:
        dict1[colour]=number
print(dict1)

```

(e)

```

n=eval(input("Enter the elements as list:"))
dict1={"ODD":0,"EVEN":0}
for num in n:
    if num%2==0:
        dict1["EVEN"]+=1
    else:
        dict1["ODD"]+=1
print(dict1)

```

Screenshot of Output:

(a)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/python/lab_10/a1.py"
Enter a string:akshara user
{'a': 3, 'k': 1, 's': 2, 'h': 1, 'r': 2, ' ': 1, 'u': 1, 'e': 1}
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code>
```

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/python/lab_10/a2.py"
Enter a string:akshara
{'a': 3, 'k': 1, 's': 1, 'h': 1, 'r': 1}
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code>
```

(b)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/python/lab_10/b.py"
Enter the list of words["akshara","btech","aids"]
{'akshara': '-----', 'btech': '-----', 'aids': '-----'}
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code>
```

©

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/python/lab_10/c.py"
Enter the number of students:3
Enter the details of student 1
Maths54
English:65
Physics89
Chemistry23
Computer41
Enter the details of student 2
Maths56
English:78
Physics46
Chemistry65
Computer23
Enter the details of student 3
Maths56
English:12
Physics78
Chemistry96
Computer63
{'student1': {'Marks': [[54, 65, 89, 23, 41]], 'Average': 54.4, 'Rank': 2}, 'student2': {'Marks': [[56, 78, 46, 65, 23]], 'Average': 53.6, 'Rank': 3}, 'student3': {'Marks': [[56, 12, 78, 96, 63]], 'Average': 61.0, 'Rank': 1}}
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code>
```

(d)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/python/lab_10/d.py"
{'yellow': 4, 'blue': 6, 'red': 1}
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code>
```

(e)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> & C:/Users/aksha/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/aksha/Desktop/AKSHARA/SEM 2/vs code/python/lab_10/e.py"
Enter the elements as list:[45,34,6,78,98,97,45]
{'ODD': 3, 'EVEN': 4}
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code>
```

Result:

Thus, programs have been written and executed to explore dictionaries in Python.

Ex. 11

EXPLORING FILE OPERATIONS

Date:15/04/24

Aim:

To explore file operations in Python by writing programs for the following:

- a. Create a new text file and write 10 lines into it.
- b. Read the text file created above and count the number of words, characters, and lines in it. Find the frequency of occurrence of each word and character.
- c. Read only the first 2 lines of the text file created above and print them. Then skip the next 4 characters and print the remaining characters in the line. Then, skip 3 lines and print the remaining lines.
- d. Create a text file with several lines. Compute the unigram, bigram, and trigram probabilities (that is, probability of occurrence of each word, pair of words, and combination of 3 words). Remove all punctuation marks from the text before computing the probabilities. Store the computed probabilities in dictionaries and display them.

Algorithm:

(a)

STEP 1: Define a long multi-line string variable `text` with information about the Chambal River.

STEP 2: Open a new file named "text" in write mode.

STEP 3: Write the contents of the `text` variable to the file.

STEP 4: Close the file to ensure all data is saved and resources are properly released.

(b)

STEP 1: Initialize variables to track the number of words, characters, and create dictionaries to hold word and character frequencies.

STEP 2: Open the file "text" in read mode.

STEP 3: Iterate over each line in the file, and for each line, increment the line count.

STEP 4: Split each line into words, update the total number of words, and calculate the number of characters in each word.

STEP 5: Update the word frequency dictionary for each word and character frequency dictionary for each character in the words.

STEP 6: Print the total number of lines, words, and characters, and display the frequency of each word and character from the file.

(c)

STEP 1: Open the file named "text" in read mode.

STEP 2: Read and print the first two lines from the file.

STEP 3: Read the next line, skip the first four characters, and then print the remaining part of this line.

STEP 4: Skip the next three lines in the file.

STEP 5: Print all remaining lines in the file after the skipped lines.

(d)

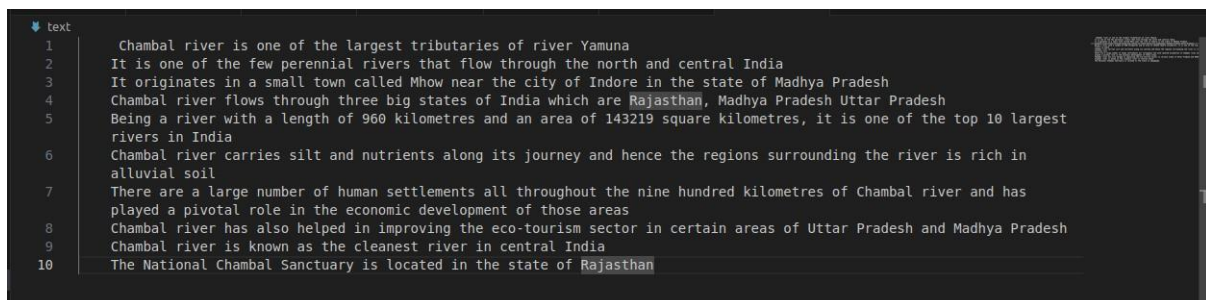
STEP 1: Open the file named "text" and read its contents into a variable. Split the content into words to create a list of words.

STEP 2: Create a list of unique words (unigrams) and calculate the probability of each word appearing in the text. Print the unigram probabilities.

STEP 3: Construct a list of consecutive word pairs (bigrams) from the text. Calculate and print the probability of each bigram occurring based on the frequency of the first word in the bigram.

STEP 4: Generate a list of consecutive word triplets (trigrams) from the text. Calculate and print the probability of each trigram based on the frequency of the corresponding bigram.

Text file:



```
1 Chambal river is one of the largest tributaries of river Yamuna
2 It is one of the few perennial rivers that flow through the north and central India
3 It originates in a small town called Mhow near the city of Indore in the state of Madhya Pradesh
4 Chambal river flows through three big states of India which are Rajasthan, Madhya Pradesh Uttar Pradesh
5 Being a river with a length of 960 kilometres and an area of 143219 square kilometres, it is one of the top 10 largest
6 rivers in India
7 Chambal river carries silt and nutrients along its journey and hence the regions surrounding the river is rich in
8 alluvial soil
9 There are a large number of human settlements all throughout the nine hundred kilometres of Chambal river and has
10 played a pivotal role in the economic development of those areas
Chambal river has also helped in improving the eco-tourism sector in certain areas of Uttar Pradesh and Madhya Pradesh
Chambal river is known as the cleanest river in central India
The National Chambal Sanctuary is located in the state of Rajasthan
```

Program:

(a)

```
text='' Chambal river is one of the largest tributaries of river Yamuna

It is one of the few perennial rivers that flow through the north and central
India
It originates in a small town called Mhow near the city of Indore in the state
of Madhya Pradesh
Chambal river flows through three big states of India which are Rajasthan,
Madhya Pradesh Uttar Pradesh Being a river with a length of 960 kilometres and
an area of 143219 square kilometres, it is one of the top 10 largest rivers in
India
Chambal river carries silt and nutrients along its journey and hence the regions
surrounding the river is rich in alluvial soil
There are a large number of human settlements all throughout the nine hundred
kilometres of Chambal river and has played a pivotal role in the economic
development of those areas
Chambal river has also helped in improving the eco-tourism sector in certain
areas of Uttar Pradesh and Madhya Pradesh
Chambalriver is known as the cleanest river in central India
```

```
The National Chambal Sanctuary is located in the state of Rajasthan'''
```

```
f=open("text","w")
f.write(text)
f.close()
```

(b)

```
num_lines=0
num_words=0
num_char=0

word_freq={}
char_freq={}

with open("text","r") as file:
    for line in file:
        num_lines+=1
        words=line.split()
        num_words=len(words)
        for word in words:
            num_char+=len(word)
            if word in word_freq.keys():
                word_freq[word]+=1
            else:
                word_freq[word]=1
                for char in word:
                    if char in char_freq.keys():
                        char_freq[char]+=1
                    else:
                        char_freq[char]=1
print("Number of lines in the file:",num_lines) print("Number of
words:",num_words) print("Number of characters:",num_char) print("Frequency of
occurrence of words:",word_freq) print()
print("Frequency of occurrence of characters",char_freq)
```

(c)

```
with open("text","r") as file:

    for i in range(2):
        print(file.readline())
        remaining_chars=file.readline()[4:]
        print(remaining_chars)
    for i in range(3):
        file.readline()
    for line in file:
        print(line)
```

```

unigram=[]
bigram=[]
trigram=[]
uni_prob={}
bi_prob={}
tri_prob={}

f=open("text","r")
file=f.read()
words=file.split()
num_words=len(words)

unigram=[word for word in set(words)]
uni_prob={word:format(words.count(word)/num_words,'.3f') for word in unigram}
print(uni_prob)

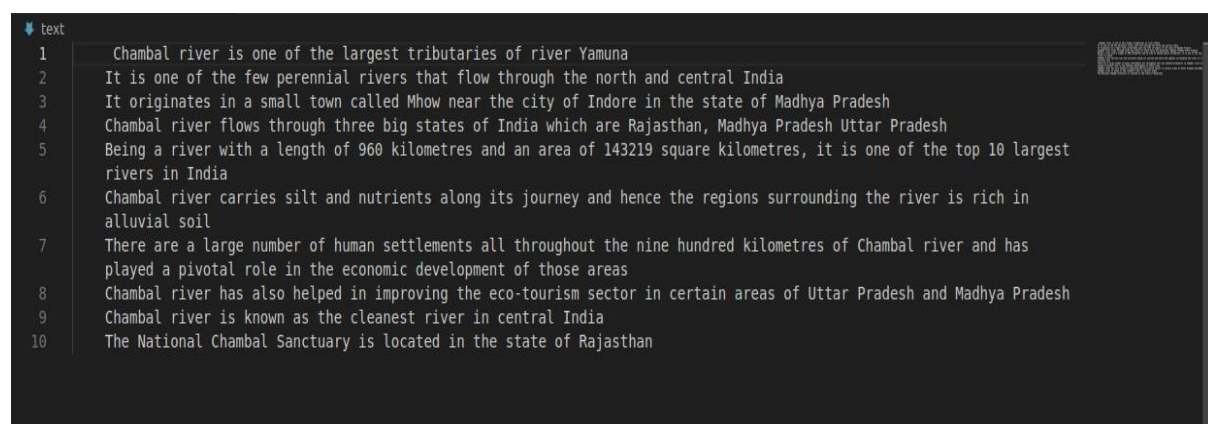
bigram=[(words[i],words[i+1]) for i in range(len(words)-1)]
bi_prob={(w1,w2):format(bigram.count((w1,w2))/words.count(w1),'.3f') for w1,w2
in set(bigram)}
print(bi_prob)

trigram=[(words[i],words[i+1],words[i+2]) for i in range(len(words)-2)]
tri_prob={(w1,w2,w3):format(trigram.count((w1,w2,w3))/bigram.count((w1,w2)),'.3f')
for w1,w2,w3 in set(trigram)}
print(tri_prob)

```

Screenshot of Output:

(a)



The screenshot shows a text editor window with a file named 'text'. The file contains 10 lines of text describing the Chambal river. The text is as follows:

- 1 Chambal river is one of the largest tributaries of river Yamuna
- 2 It is one of the few perennial rivers that flow through the north and central India
- 3 It originates in a small town called Mhow near the city of Indore in the state of Madhya Pradesh
- 4 Chambal river flows through three big states of India which are Rajasthan, Madhya Pradesh Uttar Pradesh
- 5 Being a river with a length of 960 kilometres and an area of 143219 square kilometres, it is one of the top 10 largest rivers in India
- 6 Chambal river carries silt and nutrients along its journey and hence the regions surrounding the river is rich in alluvial soil
- 7 There are a large number of human settlements all throughout the nine hundred kilometres of Chambal river and has played a pivotal role in the economic development of those areas
- 8 Chambal river has also helped in improving the eco-tourism sector in certain areas of Uttar Pradesh and Madhya Pradesh
- 9 Chambal river is known as the cleanest river in central India
- 10 The National Chambal Sanctuary is located in the state of Rajasthan

(b)

```
ai_ds_al@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$ /bin/python3 /home/ai_ds_al/Desktop/akshara/b.py
Number of lines in the file: 10
Number of words: 11
Number of characters: 859
Frequency of occurrence of words: {'Chambal': 7, 'river': 10, 'is': 6, 'one': 3, 'of': 14, 'the': 13, 'largest': 2, 'tributaries': 1, 'Yamuna': 1, 'It': 2, 'few': 1, 'perenni
al': 1, 'rivers': 2, 'that': 1, 'flow': 1, 'through': 2, 'north': 1, 'and': 6, 'central': 2, 'India': 4, 'originates': 1, 'in': 9, 'a': 5, 'small': 1, 'town': 1, 'called': 1
, 'Mhow': 1, 'near': 1, 'city': 1, 'Indore': 1, 'state': 2, 'Madhya': 3, 'Pradesh': 5, 'flows': 1, 'three': 1, 'big': 1, 'states': 1, 'which': 1, 'are': 2, 'Rajasthan': 1,
'Uttar': 2, 'Being': 1, 'with': 1, 'length': 1, '960': 1, 'kilometres': 2, 'an': 1, 'area': 1, '143219': 1, 'square': 1, 'kilometres': 1, 'it': 1, 'top': 1, '10': 1, 'carri
es': 1, 'silt': 1, 'nutrients': 1, 'along': 1, 'its': 1, 'journey': 1, 'hence': 1, 'regions': 1, 'surrounding': 1, 'rich': 1, 'alluvial': 1, 'soil': 1, 'There': 1, 'large':
1, 'number': 1, 'human': 1, 'settlements': 1, 'all': 1, 'throughout': 1, 'nine': 1, 'hundred': 1, 'has': 2, 'played': 1, 'pivotal': 1, 'role': 1, 'economic': 1, 'development
': 1, 'those': 1, 'areas': 2, 'also': 1, 'helped': 1, 'improving': 1, 'eco-tourism': 1, 'sector': 1, 'certain': 1, 'known': 1, 'as': 1, 'cleanest': 1, 'The': 1, 'National':
1, 'Sanctuary': 1, 'located': 1, 'Rajasthan': 1}

Frequency of occurrence of characters {'C': 7, 'h': 54, 'a': 94, 'm': 19, 'b': 10, 'l': 41, 'r': 73, 'i': 62, 'v': 16, 'e': 95, 's': 48, 'o': 51, 'n': 60, 'f': 17, 't': 68, '
g': 14, 'u': 17, 'Y': 1, 'I': 7, 'w': 8, 'p': 7, 'd': 27, 'c': 16, 'M': 4, 'y': 7, 'P': 5, 'R': 2, 'j': 3, ',': 2, 'U': 2, 'B': 1, '9': 2, '6': 1, '0': 2, 'k': 4, '1': 3, '4
': 1, '3': 1, '2': 1, 'q': 1, 'T': 2, '-': 1, 'N': 1, 'S': 1}
ai_ds_al@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$
```

(c)

```
ai_ds_al@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$ /bin/python3 /home/ai_ds_al/Desktop/akshara/c.py
Chambal river is one of the largest tributaries of river Yamuna

It is one of the few perennial rivers that flow through the north and central India

It originates in a small town called Mhow near the city of Indore in the state of Madhya Pradesh

There are a large number of human settlements all throughout the nine hundred kilometres of Chambal river and has played a pivotal role in the economic development of th
ose areas

Chambal river has also helped in improving the eco-tourism sector in certain areas of Uttar Pradesh and Madhya Pradesh

Chambal river is known as the cleanest river in central India

The National Chambal Sanctuary is located in the state of Rajasthan
ai_ds_al@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$
```

(d)

```
ai_ds_al@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$ /bin/python3 /home/ai_ds_al/Desktop/akshara/d.py
{'areas': '0.024', 'certain': '0.024', 'Chambal': '0.073', 'The': '0.024', 'has': '0.024', 'Uttar': '0.024', 'Madhya': '0.024', 'sector': '0.024', 'river': '0.073', 'central': '0.024', 'in
': '0.098', 'state': '0.024', 'National': '0.024', 'is': '0.049', 'located': '0.024', 'improving': '0.024', 'helped': '0.024', 'Sanctuary': '0.024', 'eco-tourism': '0.024', 'known': '0.024
', 'of': '0.049', 'as': '0.024', 'Rajasthan': '0.024', 'the': '0.073', 'cleanest': '0.024', 'and': '0.024', 'India': '0.024', 'Pradesh': '0.049', 'also': '0.024'}
({'Chambal', 'river'): '0.667', ('also', 'helped'): '1.000', ('Pradesh', 'and'): '0.500', ('Sanctuary', 'is'): '1.000', ('improving', 'the'): '1.000', ('state', 'of'): '1.000', ('The', 'Na
tional'): '1.000', ('National', 'Chambal'): '1.000', ('helped', 'in'): '1.000', ('river', 'has'): '0.333', ('central', 'India'): '1.000', ('located', 'in'): '1.000', ('is', 'known'): '0.50
0', ('Chambal', 'Sanctuary'): '0.333', ('as', 'the'): '1.000', ('Pradesh', 'Chambal'): '0.500', ('cleanest', 'river'): '1.000', ('of', 'Rajasthan'): '0.500', ('certain', 'areas'): '1.000',
('has', 'also'): '1.000', ('in', 'central'): '0.250', ('Madhya', 'Pradesh'): '1.000', ('river', 'in'): '0.333', ('the', 'cleanest'): '0.333', ('the', 'state'): '0.333', ('Uttar', 'Pradesh
'): '1.000', ('river', 'is'): '0.333', ('in', 'the'): '0.250', ('in', 'improving'): '0.250', ('and', 'Madhya'): '1.000', ('the', 'eco-tourism'): '0.333', ('of', 'Uttar'): '0.500', ('known', '
as'): '1.000', ('eco-tourism', 'sector'): '1.000', ('India', 'The'): '1.000', ('in', 'certain'): '0.250', ('sector', 'in'): '1.000', ('areas', 'of'): '1.000', ('is', 'located'): '0.500'
})
({'the', 'state', 'of'): '1.000', ('Chambal', 'Sanctuary', 'is'): '1.000', ('Pradesh', 'and', 'Madhya'): '1.000', ('state', 'of', 'Rajasthan'): '1.000', ('the', 'cleanest', 'river'): '1.00
0', ('Madhya', 'Pradesh', 'Chambal'): '1.000', ('India', 'The', 'National'): '1.000', ('helped', 'in', 'improving'): '1.000', ('Chambal', 'river', 'has'): '0.500', ('river', 'in', 'central
'): '1.000', ('improving', 'the', 'eco-tourism'): '1.000', ('river', 'has', 'also'): '1.000', ('as', 'the', 'cleanest'): '1.000', ('Pradesh', 'Chambal', 'river'): '1.000', ('cleanest', 'ri
ver', 'in'): '1.000', ('also', 'helped', 'in'): '1.000', ('areas', 'of', 'Uttar'): '1.000', ('The', 'National', 'Chambal'): '1.000', ('Uttar', 'Pradesh', 'and'): '1.000', ('known', 'as', '
the'): '1.000', ('the', 'eco-tourism', 'sector'): '1.000', ('in', 'central', 'India'): '1.000', ('central', 'India', 'The'): '1.000', ('in', 'improving', 'the'): '1.000', ('located', 'in',
'the'): '1.000', ('National', 'Chambal', 'Sanctuary'): '1.000', ('eco-tourism', 'sector', 'in'): '1.000', ('and', 'Madhya', 'Pradesh'): '1.000', ('Sanctuary', 'is', 'located'): '1.000', ('
Chambal', 'river', 'is'): '0.500', ('is', 'located', 'in'): '1.000', ('in', 'certain', 'areas'): '1.000', ('sector', 'in', 'certain'): '1.000', ('has', 'also', 'helped'): '1.000', ('in', '
the', 'state'): '1.000', ('of', 'Uttar', 'Pradesh'): '1.000', ('certain', 'areas', 'of'): '1.000', ('is', 'known', 'as'): '1.000', ('river', 'is', 'known'): '1.000'}
ai_ds_al@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$
```

Result:

Thus, programs have been written and executed to explore file operations in Python.

Ex. 12**EXPLORING COMMAND-LINE ARGUMENTS****Date: 22/04/24****Aim:**

To explore command-line arguments in Python by writing programs for the following:

- a. Given a sentence as a command-line argument, compute the probability of the sentence being valid based on bigram probabilities computed in the previous exercise.
- b. Given a file name as a command line argument, check if the file exists. If the file exists, read the contents of the file and display it. If the file does not exist, create the file and add some content to it. (Use try ... except ... else blocks.)
- c. Given an integer as a command-line argument, print the list of prime numbers less than the given integer.

Algorithm:

(a)

STEP 1: Remove punctuations from the entire text file to normalize the data. Split the text into individual words.

STEP 2: Create a list of bigrams, which are consecutive word pairs, from the list of words.

STEP 3: Calculate the frequency of each bigram relative to the appearance of its first word, and store these probabilities in a dictionary. Each bigram probability is calculated as the count of the bigram divided by the count of its first word.

STEP 4: Read the input sentence, strip it of punctuations, and split it into words.

STEP 5: Initialize a variable `prob` to 1. For each consecutive word pair (bigram) in the sentence, multiply `prob` by the bigram's probability from the dictionary if it exists. If a bigram does not exist in the dictionary, multiply `prob` by 0, effectively setting the probability of the entire sequence to zero.

STEP 6: Output the computed probability of the sentence. This probability represents the likelihood of the sentence's word sequence based on the bigram frequencies in the provided text.

(b)

STEP 1: Retrieve the filename from the command-line arguments.

STEP 2: Attempt to open and read the contents of the specified file.

STEP 3: If the file does not exist (FileNotFoundError), create the file and write "some contents" to it.

STEP 4: Display the file contents if the file is read successfully.

(c)

STEP 1: Retrieve the upper limit N from the command-line arguments.

STEP 2: Iterate through each number from 2 to N .

STEP 3: Assume each number is prime.

STEP 4: Check if the current number is divisible by any smaller number (from 2 to the number minus one). If divisible, mark the number as not prime and break the loop.

STEP 5: If the number is prime, print it.

Program:

(a)

```
import sys
bigram=[]

bi_prob={}
punctuations = ['.', ',', '?', '!', ':', ';', '"', "'", '(', ')', '[', ']',
'{' , '}', '-', '—', '/', '&', '*', '$', '%', '#', '@', '+', '=', '~', '^',
'\\', '|', '•', '†', '“', '”', '”']

with open("text.txt",encoding="utf8") as text:
    file=text.read()
    for p in punctuations:
        file=file.replace(p,'')
words= file.split()
bigram=[(words[i].lower(),words[i+1].lower()) for i in range(len(words)-1)]
bi_prob={(w1,w2):format(bigram.count((w1,w2))/words.count(w1),'.3f') for w1,w2
in set(bigram)}

prob=1
sentence=sys.argv[1]

for p in punctuations:
    sentence=sentence.replace(p,"")
sentence_words=sentence.split()

for i in range(len(sentence_words) - 1):
    pair = (sentence_words[i].lower(), sentence_words[i+1].lower())
    if pair in bi_prob.keys():
        prob *= bi_prob[pair]
```

```
print(prob)
```

(b)

```
import sys
read_file=sys.argv[1]

try:
    with open(read_file,"r") as file:
        print(file.read())
except FileNotFoundError:
    with open(read_file,"w") as file:
        file.write("some contents")
```

(c)

```
import sys
N = int(sys.argv[1])

for num in range(2, N + 1):
    prime = True

    for i in range(2, num):
        if num % i == 0:
            prime = False
            break

    if prime:
        print(num, end=' ')
```

Screenshot of Output:

(a)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> python3 a.py "youll probably wont know"
0.500
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> █
```


(b)

```
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> python3 b.py text.txt
```

The writing process of preparation, writing, and revisions applies to every essay or paper, but the time and effort spent on each stage depends on the type of essay.

For example, if you've been assigned a five-paragraph expository essay for a high school class, you'll probably spend the most time on the writing stage; for a college-level argumentative essay, on the other hand, you'll need to spend more time researching your topic and developing an original argument before you start writing.

(c)

```
TypeError: can only concatenate str (not 'int') to str
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> python3 c.py 25
2 3 5 7 11 13 17 19 23
PS C:\Users\aksha\Desktop\AKSHARA\SEM 2\vs code> █
```

Result:

Thus, programs have been written and executed to explore command-line arguments in Python.