Graphic Era
HILL UNIVERSITY
Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

Practical File

Python Lab (PBC-401)

BCA Fourth Semester

Session 2023-2024

Submitted to:

Dr. Janmejay Pant

Assistant Professor

School of Computing

Graphic Era Hill University

Bhimtal Campus

Submitted by:

Harshita Chauhan

BCA

Section A

Roll No- 2271124

Question 1:- WAP to demonstrate different data types.

Answer 1:-

```
a = 13
b = 3.1425
c = 'Harshita'
d = {'abc' , 'def' , 'ghi'}
e = ['123' , '456' , '789']
f = False


print(type(a))
print(type(b))
print(type(c))
print(type(d))
print(type(e))
print(type(f))
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python Data.py
<class 'int'>
<class 'float'>
<class 'str'>
<class 'set'>
<class 'list'>
<class 'bool'>
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 2:- WAP to perform different arithmetic operations on number in python.

Answer 2:-

```
num1 = int(input("Enter 1st number "))
num2 = int(input("Enter 2nd number "))
add = num1+num2
sub = num1-num2
mul = num1*num2
div = num1/num2
print(f"Addition is :{add}")
print(f"Subtraction is :{sub}")
print(f"Multiplication is :{mul}")
print(f"Division is :{div}")
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python Arithematic.py
Enter 1st number 5
Enter 2nd number 2
Addition is :7
Subtraction is :3
Multiplication is :10
Division is :2.5
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 3:- WAP to compute distance between 2 points taking input from the user.

Answer3:-

```
import math
x1 = int(input("Enter the value for x1 : "))
x2 = int(input("Enter the value for x2 : "))
y1 = int(input("Enter the vakue for y1 : "))
y2 = int(input("Enter the value for y2 : "))

d = math.sqrt((x2-x1)**2+(y2-y1)**2)
print(f"Distance is {d}")
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python Distance.py
Enter the value for x1 : 2
Enter the value for x2 : 3
Enter the vakue for y1 : 4
Enter the value for y2 : 5
Distance is 1.4142135623730951
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 4:- WAP to add two numbers taking input from command line arguments and print the arguments.

Answer 4:-

```
import sys
a = sys.argv[1]
b = sys.argv[2]
c = int(a) + int(b)
print(f"Sum is {c}")
print(sys.argv)
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python CommandLine.py 2 3
Sum is 5
['CommandLine.py', '2', '3']
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 5:- Program on List.
Student=[556,"Mohit",84,96,84,75,84]

Create a List containing the following items and perform the following operations:-

(1) Access 0th element (2) Access 0th to 1st elements (3) Access 2nd to end of list elements (4) Access starting to 2nd elements (5) Access starting to ending elements (6) Access last index value (7) Access elements in reverse order

Answer 5:-

```
student = [556,"Mohit",84,96,84,75,84]
print(f"1st element of the list is {student[0]}")
print(f"0th to 1st element of the list is {student[0:2]}")
print(f"2nd to the end of the List {student[1:len(student)]}")
print(f"Starting to 2nd elements {student[1:3]}")
print(f"Starting to ending elements {student}")
print(f"Last index value {student[-1]}")
print(f"Reverse of the List {student[::-1]}")
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python List.py
1st element of the list is 556
0th to 1st element of the list is [556, 'Mohit']
2nd to the end of the List ['Mohit', 84, 96, 84, 75, 84]
Starting to 2nd elements ['Mohit', 84]
Starting to ending elements [556, 'Mohit', 84, 96, 84, 75, 84]
Last index value 84
Reverse of the List [84, 75, 84, 96, 84, 'Mohit', 556]
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 6:- WAP a program to create a dictionary in python as follows:-

d1 = {'First':'Sunday', 1:'Monday',2:'Tuesday'}

Answer 6:-

```
d1 = {'First':'Sunday', 1:'Monday', 2:'Tueday',3:'Wednesday'}.
print(d1.get(3))
print(d1.keys())
print(d1.values())
print(d1.get('First'))
d1[5] = 'Saturday'
print(d1.values())
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python Dict.py
Wednesday
dict_keys(['First', 1, 2, 3])
dict_values(['Sunday', 'Monday', 'Tueday', 'Wednesday'])
Sunday
dict_values(['Sunday', 'Monday', 'Tueday', 'Wednesday', 'Saturday'])
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 7:- WAP for dataframes in python as follows:-

std = {"id":[10,20,30,40,50], "marks":[51,61,71,81,91], "subject":['c','c++','java','python','os']}

Answer 7:-

```
import pandas as pd

std = {"id":[10,20,30,40,50], "marks":[51,61,71,81,91],
"subject":['c','c++','java','python','os']}

df = pd.DataFrame(std)

print(df , "\n")

df = pd.DataFrame(std,index=['a','b','c','d','e'])

print(df , "\n")

df = pd.DataFrame(std,columns=['id','marks'])

print(df)
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python Dataframe.py
   id  marks subject
0  10     51       c
1  20     61     c++
2  30     71    java
3  40     81  python
4  50     91      os

   id  marks subject
a  10     51       c
b  20     61     c++
c  30     71    java
d  40     81  python
e  50     91      os

   id  marks
0  10     51
1  20     61
2  30     71
3  40     81
4  50     91
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 8:- WAP of Set in python and perform the following operations:-

(1) Union (2) Intersection (3) XOR (4) Difference

Answer 8:-

```
s1 = {1,2,3,4,'Harshita'}

s2 = set([1,2,3,4])

print(f"Union :{s1|s2}")

print(f"Intersection :{s1&s2}")

print(f"Difference :{s1-s2}")

print(f"XOR :{s1^s2}")
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python Set.py
Union :{1, 2, 3, 4, 'Karan'}
Intersection :{1, 2, 3, 4}
Difference :{'Karan'}
XOR :{'Karan'}
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 9:- WAP to check whether a number is even or odd number should be taken from user.

Answer 9:-

```
n = int(input("Enter a number "))
if(n%2 == 0):
    print("Even")
else:
    print("Odd")
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python EvenOdd.py
Enter a number 12
Even
PS C:\Users\himan\OneDrive\Desktop\Python> python EvenOdd.py
Enter a number 15
Odd
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 10:- WAP to get marks of subjects from the user and calculate the average marks score and calculate the grade obtain based on the average marks :91-100=a/81-90=b/71-80=c/61-70=d/51-60=e/<=50=f

Answer 10:-

```
marks = []
sum = 0
print("Enter 5 marks")
for i in range (5):
        mark = float(input())
        marks.append(mark)
        sum = sum + marks[i]
avg = sum/5
print(f"Average is :{avg}")
if (91 <= avg <= 100):
    print("Grade is: A")
elif (81 <= avg <= 90) :
    print("Grade is: B")
elif (71 <= avg <= 80) :
    print("Grade is: C")
elif (61 <= avg <= 70) :
```

```
    print("Grade is: D")
elif (51 <= avg <= 60) :
    print("Grade is: E")
else:
    print("Grade is: F")
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python Grade.py
Enter 5 marks
92
95
91
88
82
Average is :89.6
Grade is: B
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 11:- WAP to print one to n numbers n should be entered by the user.

Answer 11:-

```
n = int(input("Enter a number "))
print("Using for loop")
for i in range(1,n+1):
        print(i)

print("Using while loop")
i=1
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python n.py
Enter a number 10
Using for loop
1
2
3
4
5
6
7
8
9
10
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 12:- WAP to print even number from 1 to 50.

Answer 12:-

```
for i in range(1,52):
    if(i%2==0):
        print(i)
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python Even.py
2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 13:- WAP to find the sum of n natural numbers using while loop.

Answer 13:-

```
n = int(input("Enter a number "))
i = 1
sum = 0
while(i<=n):
    sum = sum + i
    i=i+1
print(f"Sum of {n} numbers are {sum}")
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python SumNatural2.py
Enter a number 100
Sum of 100 numbers are 5050
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 14:- Find the sum of n numbers entered by the user using for loop.

Answer 14:-

```
n = int(input("Enter a number "))
sum = 0
for i in range(1,n+1):
    sum = sum+i
print(f"Sum of {n} numbers are {sum}")
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python SumNatural.py
Enter a number 100
Sum of 100 numbers are 5050
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 15:- Find the sum of 2 matrices using for loop.

Answer 15:-

```
m1 = [[1,2,3],[4,5,6],[7,8,9]]
m2 = [[11,22,33],[44,55,66],[77,88,99]]
result = [[0, 0, 0] for _ in range(len(m1))]
for i in range(len(m1)):
    for j in range(len(m1[0])):
        result[i][j] = m1[i][j] + m2[i][j]
print("Sum of the matrices is:")
for row in result:
    print(row)
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python SumMatrice.py
Sum of the matrices is:
[12, 24, 36]
[48, 60, 72]
[84, 96, 108]
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 16:- Print the pattern.

1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

Answer 16:-

```
print("Pattern is ")
for i in range(1, 5):
    for j in range(i):
        print(i ,end=" ")
    print()
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python Pattern.py
Pattern is
1
2 2
3 3 3
4 4 4 4
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 17:- Wap to implement a simple calculator that performs basic mathematical operations it takes three inputs from the user at first the user is asked to enter two numbers after that the user is asked to enter operator based on which airthmetic operation is performed.

Answer 17:-

```
num1 = int(input("Enter 1st number : "))
num2 = int(input("Enter 2nd number : "))
ch = (input("Enter an operator : "))
ans = 0
if (ch == '+') :
    ans = num1 + num2
    print(f"Addition is : {ans}")
elif(ch == '-') :
    ans = num1 - num2
    print(f"Subtraction is : {ans}")
elif(ch == '*') :
    ans = num1 * num2
    print(f"Multiplication is : {ans}")
elif(ch == '/') :
    ans = num1 / num2
```

```
        print(f"Division is : {ans}")
else :
        print("Wrong Operation character")
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python Calculator.py
Enter 1st number : 25
Enter 2nd number : 5
Enter an operator : /
Division is : 5.0
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 18:- Wap to check whether a character enter by the user is in upper case or the lower case.

Answer 18:-

```
ch = input("Enter a character :")
if(ch.isupper()) :
    print("Character is in Upper Case")
elif(ch.islower()) :
    print("Character is in Lower Case")
```

Output:-

```
PS C:\Users\himan\OneDrive\Desktop\Python> python Case.py
Enter a character :A
Character is in Upper Case
PS C:\Users\himan\OneDrive\Desktop\Python> python Case.py
Enter a character :a
Character is in Lower Case
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 19:- WAP that asks the user to enter a number. If the number is greater than or equal to 100, the program should determine whether the number is even or odd. If the number is less than 100, the program should only print the message "The number is less than 100"

Answer 19:-

```
n = int(input("Enter a number :"))
if(n >= 100):
    if(n %2 == 0):
        print("Number is greater than 100 and even.")
    else :
        print("Number is greater than 100 and odd.")
else :
    print("Number is less than 100")
```

Output:-

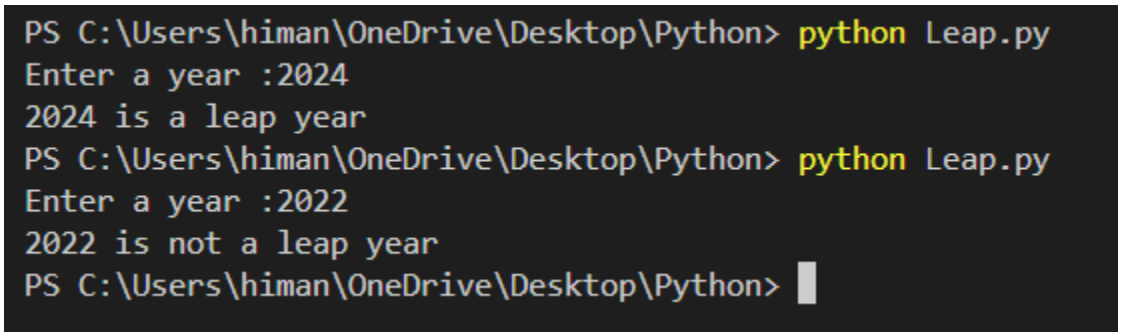
```
PS C:\Users\himan\OneDrive\Desktop\Python> python Check100.py
Enter a number :102
Number is greater than 100 and even.
PS C:\Users\himan\OneDrive\Desktop\Python> python Check100.py
Enter a number :101
Number is greater than 100 and odd.
PS C:\Users\himan\OneDrive\Desktop\Python> python Check100.py
Enter a number :95
Number is less than 100
PS C:\Users\himan\OneDrive\Desktop\Python>
```

Question 20:- Wap to check whether the given year is leap year or not:-

Answer 20:-

```
year = int(input("Enter a year :"))
if((year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)):
    print(f"{year} is a leap year")
else :
    print(f"{year} is not a leap year")
```

Output:-



Ques 19: wap to create basic calculator using module

```
# Define functions for basic arithmetic operations
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    if y == 0:
        return "Error! Division by zero."
    else:
        return x / y

def main():
    try:
        # Take input from the user for two numbers
        num1 = float(input("Enter the first number: "))
        num2 = float(input("Enter the second number: "))

        # Display the menu of operations
        print("\nOperations:")
        print("1. Addition")
        print("2. Subtraction")
```

```
        print("3. Multiplication")
        print("4. Division")

        # Take input from the user for the choice of operation
        choice = input("Enter the operation (1/2/3/4): ")

        # Perform the selected operation and display the result
        if choice == '1':
            print("Result:", add(num1, num2))
        elif choice == '2':
            print("Result:", subtract(num1, num2))
        elif choice == '3':
            print("Result:", multiply(num1, num2))
        elif choice == '4':
            print("Result:", divide(num1, num2))
        else:
            print("Invalid choice. Please enter a valid operation.")
    except ValueError:
        print("Please enter valid numeric values.")

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

OUTPUT :

```
Enter the first number:  20
Enter the second number:  2

Operations:
1. Addition
2. Subtraction
3. Multiplication
```

4. Division
Enter the operation (1/2/3/4): 3
Result: 40.0

Ques 20: create a txt file intern.txt and ask the user to write a single line of text by user input

```
def main():
    try:
        # Open the file in write mode ('w')
        with open("intern.txt", "w") as file:
            # Ask the user to write a single line of text
            line = input("Write a single line of text: ")

            # Write the user's input to the file
            file.write(line)

        print("Text has been written to 'intern.txt'.")
    except Exception as e:
        print("An error occurred:", str(e))

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

OUTPUT :

```
Write a single line of text:  HELLO THIS IS A TEXT PAGE
Text has been written to 'intern.txt'.
```

Ques 21: create a txt file myfile.txt and ask the user to write separate 3 lines with 3 inputs statements from the user

```
def main():
    try:
        # Open the file in write mode ('w')
        with open("myfile.txt", "w") as file:
            # Ask the user to write three separate lines
            for i in range(3):
                line = input("Enter line {}: ".format(i + 1))
                file.write(line + "\n")

        print("Three lines have been written to 'myfile.txt'.")
    except Exception as e:
        print("An error occurred:", str(e))

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

OUTPUT :

```
Enter line 1:  Hello, this is line 1.
Enter line 2:  This is line 2.
Enter line 3:  This is line 3.
Three lines have been written to 'myfile.txt'.
```

Ques 22: wap to read the content of both the file created in above program and merge the content intol merge.txt

```
def merge_files():
    try:
        # Open the first file (intern.txt) in read mode ('r')
        with open("intern.txt", "r") as file1:
            # Read the contents of the first file
            content1 = file1.read()

        # Open the second file (myfile.txt) in read mode ('r')
        with open("myfile.txt", "r") as file2:
            # Read the contents of the second file
            content2 = file2.read()

        # Merge the contents of both files
        merged_content = content1 + "\n" + content2

        # Write the merged content to a new file (merge.txt)
        with open("merge.txt", "w") as merge_file:
            merge_file.write(merged_content)

        print("Contents of both files have been merged into 'merge.txt'.")
    except Exception as e:
        print("An error occurred:", str(e))

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

OUTPUT :

Contents of both files have been merged into 'merge.txt'.

Ques 23: count the total number of uppercase and lowercase and digit used in merge.txt

```
def count_characters(filename):
    try:
        # Open the file in read mode ('r')
        with open(filename, "r") as file:
            # Read the contents of the file
            content = file.read()

            # Initialize counters
            uppercase_count = 0
            lowercase_count = 0
            digit_count = 0

            # Iterate through each character in the content
            for char in content:
                if char.isupper():
                    uppercase_count += 1
                elif char.islower():
                    lowercase_count += 1
                elif char.isdigit():
                    digit_count += 1

            # Print the counts
            print("Total number of uppercase letters:", uppercase_count)
            print("Total number of lowercase letters:", lowercase_count)
            print("Total number of digits:", digit_count)
```

```
    except Exception as e:
        print("An error occurred:", str(e))

if __name__ == "__main__":
    count_characters("merge.txt")
```

OUTPUT :

```
Total number of uppercase letters: 23
Total number of lowercase letters: 32
Total number of digits: 3
```

Ques 24: wap to find a given number is prime or not

```
def is_prime(number):
    if number <= 1:
        return False
    elif number <= 3:
        return True
    elif number % 2 == 0 or number % 3 == 0:
        return False
    i = 5
    while i * i <= number:
        if number % i == 0 or number % (i + 2) == 0:
            return False
        i += 6
    return True

def main():
    try:
        # Take input from the user
        num = int(input("Enter a number: "))

        # Check if the number is prime
        if is_prime(num):
            print(num, "is a prime number.")
        else:
            print(num, "is not a prime number.")
    except ValueError:
```

```
        print("Please enter a valid integer.")


if __name__ == "__main__":

    main()
```

OUTPUT :

```
Enter a number:  23
23 is a prime number.
```

Ques 25: develop a python program to print all the prime numbers within a range of numbers

```
def is_prime(number):
    if number <= 1:
        return False
    elif number <= 3:
        return True
    elif number % 2 == 0 or number % 3 == 0:
        return False
    i = 5
    while i * i <= number:
        if number % i == 0 or number % (i + 2) == 0:
            return False
        i += 6
    return True

def print_primes(start, end):
    print("Prime numbers between", start, "and", end, "are:")
    for num in range(start, end + 1):
        if is_prime(num):
            print(num, end=" ")

def main():
    try:
        # Take input from the user for the range of numbers
        start = int(input("Enter the start of the range: "))
        end = int(input("Enter the end of the range: "))
```

```
        # Print prime numbers within the range
        print_primes(start, end)
    except ValueError:
        print("Please enter valid integers for the range.")

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

OUTPUT :

```
Enter the start of the range:  1
Enter the end of the range:  50
Prime numbers between 1 and 50 are:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

Ques 26:develop a python program to find the largest and smallest number in the list

```
def find_largest_and_smallest(numbers):
    if not numbers:
        return None, None

    # Initialize the largest and smallest with the first element of the list
    largest = numbers[0]
    smallest = numbers[0]

    # Iterate through the list to find the largest and smallest numbers
    for number in numbers:
        if number > largest:
            largest = number
        if number < smallest:
            smallest = number

    return largest, smallest

def main():
    try:
        # Take input from the user for the list of numbers
        input_list = input("Enter a list of numbers separated by spaces: ")
        numbers = list(map(int, input_list.split()))

        # Find the largest and smallest numbers in the list
        largest, smallest = find_largest_and_smallest(numbers)
```

```
        if largest is not None and smallest is not None:
            print("Largest number in the list is:", largest)
            print("Smallest number in the list is:", smallest)
        else:
            print("The list is empty.")
    except ValueError:
        print("Please enter a valid list of integers.")

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

OUTPUT :

```
Enter a list of numbers separated by spaces:  10 20 50 30 40
Largest number in the list is: 50
Smallest number in the list is: 10
```

Ques 27: develop a python program to develop a calculator and perform the basic calculation based on the user input and menu should be continuity available

```
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    if y == 0:
        return "Error! Division by zero."
    else:
        return x / y

def menu():
    print("\nSelect operation:")
    print("1. Addition")
    print("2. Subtraction")
    print("3. Multiplication")
    print("4. Division")
    print("5. Exit")

def main():
    while True:
```

```
menu()

try:
    choice = input("Enter choice (1/2/3/4/5): ")

    if choice == '5':
        print("Exiting the calculator. Goodbye!")
        break

    if choice in ('1', '2', '3', '4'):
        num1 = float(input("Enter first number: "))
        num2 = float(input("Enter second number: "))

        if choice == '1':
            print(f"The result of addition is: {add(num1, num2)}")
        elif choice == '2':
            print(f"The result of subtraction is: {subtract(num1, num2)}")
        elif choice == '3':
            print(f"The result of multiplication is: {multiply(num1, num2)}")
        elif choice == '4':
            print(f"The result of division is: {divide(num1, num2)}")
    else:
        print("Invalid Input. Please choose a valid operation.")

except ValueError:
    print("Invalid input. Please enter numeric values.")
```

```
if __name__ == "__main__":

   main()
```

OUTPUT :

```
Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter choice (1/2/3/4/5):  1
Enter first number:  2
Enter second number:  3
The result of addition is: 5.0

Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter choice (1/2/3/4/5):  2
Enter first number:  5
Enter second number:  3
The result of subtraction is: 2.0

Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter choice (1/2/3/4/5):  3
Enter first number:  2
Enter second number:  3
The result of multiplication is: 6.0

Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter choice (1/2/3/4/5):  4
Enter first number:  10
Enter second number:  2
The result of division is: 5.0

Select operation:
1. Addition
2. Subtraction
3. Multiplication
```

4. Division
5. Exit
Enter choice (1/2/3/4/5): 5
Exiting the calculator. Goodbye!

Ques 28: wap to show all the arguments of a function

```
def show_arguments(*args, **kwargs):
    # Display positional arguments
    print("Positional arguments:")
    for i, arg in enumerate(args):
        print(f"arg{i + 1}: {arg}")

    # Display keyword arguments
    print("\nKeyword arguments:")
    for key, value in kwargs.items():
        print(f"{key}: {value}")

def main():
    # Example usage of the function with different arguments
    show_arguments(1, "hello", True, name="Alice", age=30, city="Wonderland")

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

OUTPUT :

```
Positional arguments:
arg1: 1
arg2: hello
arg3: True

Keyword arguments:
name: Alice
age: 30
city: Wonderland
```

[ ]:

Ques 29: wap to create a function for simple calculator

```
def simple_calculator(num1, num2, operator):
    if operator == '+':
        return num1 + num2
    elif operator == '-':
        return num1 - num2
    elif operator == '*':
        return num1 * num2
    elif operator == '/':
        if num2 == 0:
            return "Error! Division by zero."
        else:
            return num1 / num2
    else:
        return "Invalid operator!"

def main():
    while True:
        try:
            # Take input from the user
            num1 = float(input("Enter the first number: "))
            num2 = float(input("Enter the second number: "))
            operator = input("Enter an operator (+, -, *, /): ")

            # Calculate the result using the simple_calculator function
            result = simple_calculator(num1, num2, operator)
```

```
            # Print the result
            print(f"The result is: {result}")

            # Ask the user if they want to perform another calculation
            again = input("Do you want to perform another calculation? (yes/no): ").strip().lower()
            if again != 'yes':
                print("Exiting the calculator. Goodbye!")
                break
        except ValueError:
            print("Invalid input. Please enter numeric values for the numbers.")

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

OUTPUT :

```
Enter the first number:  10
Enter the second number:  5
Enter an operator (+, -, *, /):  *
The result is: 50.0
Do you want to perform another calculation? (yes/no):  YES
Enter the first number:  10
Enter the second number:  5
Enter an operator (+, -, *, /):  -
The result is: 5.0
Do you want to perform another calculation? (yes/no):  no
Exiting the calculator. Goodbye!
```

Ques 30: wap to find the middle element of the string passed and id middle element is not therir than return nothing

```
def find_middle_element(s):
    length = len(s)

    # Check if the length of the string is odd
    if length % 2 != 0:
        middle_index = length // 2
        return s[middle_index]
    else:
        return "Nothing"

def main():
    # Take input from the user
    user_input = input("Enter a string: ")

    # Find the middle element
    result = find_middle_element(user_input)

    # Print the result
    print("The middle element is:", result)

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

OUTPUT :

Enter a string: Hello The middle element is: l

Ques 31: wap by define a function to calculate mean median of array of the number

```
def calculate_mean(numbers):
    if not numbers:
        return None  # Return None if the list is empty
    return sum(numbers) / len(numbers)

def calculate_median(numbers):
    if not numbers:
        return None  # Return None if the list is empty

    sorted_numbers = sorted(numbers)
    length = len(sorted_numbers)
    middle_index = length // 2

    if length % 2 == 0:
        # If even, return the average of the middle two elements
        median = (sorted_numbers[middle_index - 1] + sorted_numbers[middle_index]) / 2
    else:
        # If odd, return the middle element
        median = sorted_numbers[middle_index]

    return median

def main():
    try:
```

```
        # Take input from the user for the list of numbers
        input_list = input("Enter a list of numbers separated by spaces: ")
        numbers = list(map(float, input_list.split()))

        # Calculate mean and median
        mean = calculate_mean(numbers)
        median = calculate_median(numbers)

        # Print the results
        if mean is not None and median is not None:
            print(f"The mean of the numbers is: {mean}")
            print(f"The median of the numbers is: {median}")
        else:
            print("The list is empty.")
    except ValueError:
        print("Please enter a valid list of numbers.")

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

OUTPUT :

```
Enter a list of numbers separated by spaces:  1 2 3 4 5
The mean of the numbers is: 3.0
The median of the numbers is: 3.0
```

Ques 32: wap to change the string to the new string where the first and the last character has been existed

```
def swap_first_last_characters(s):
    # Check if the string is empty or has only one character
    if len(s) <= 1:
        return s

    # Swap the first and last characters
    new_string = s[-1] + s[1:-1] + s[0]
    return new_string

def main():
    # Take input from the user
    user_input = input("Enter a string: ")

    # Get the new string with the first and last characters swapped
    result = swap_first_last_characters(user_input)

    # Print the result
    print("The new string is:", result)

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

OUTPUT :

```
Enter a string:  HELLO
The new string is: OELLH
```

Ques 33:create a function to print the length of the string

```
def print_string_length(s):
    # Calculate the length of the string
    length = len(s)

    # Print the length of the string
    print(f"The length of the string is: {length}")

def main():
    # Take input from the user
    user_input = input("Enter a string: ")

    # Call the function to print the length of the string
    print_string_length(user_input)

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

OUTPUT :

```
Enter a string:  HELLO
The length of the string is: 5
```

Ques 34: create a function to print the square of the number using the function

```
def print_square_of_number(n):
    # Calculate the square of the number
    square = n * n

    # Print the square of the number
    print(f"The square of {n} is: {square}")

def main():
    try:
        # Take input from the user
        user_input = float(input("Enter a number: "))

        # Call the function to print the square of the number
        print_square_of_number(user_input)
    except ValueError:
        print("Invalid input. Please enter a valid number.")

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

OUTPUT :

```
Enter a number:  5
The square of 5.0 is: 25.0
```

Ques 35: write a function to take name as a function as input

```
def print_name():
    # Take name input from the user
    name = input("Enter your name: ")

    # Print the name
    print(f"Hello, {name}!")

def main():
    # Call the function to take name input and print it
    print_name()

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

OUTPUT :

```
Enter your name:  SAURABH
Hello, SAURABH!
```

Ques 36: wap to find the factorial of a no

```
def factorial(n):
    if n < 0:
        return "Factorial is not defined for negative numbers."
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result

def main():
    try:
        # Take input from the user
        user_input = int(input("Enter a number to find its factorial: "))

        # Calculate the factorial using the factorial function
        result = factorial(user_input)

        # Print the result
        print(f"The factorial of {user_input} is: {result}")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")

if __name__ == "__main__":
```

```
main()
```

OUTPUT :

```
Enter a number to find its factorial:  5
The factorial of 5 is: 120
```

Ques 37:write a pandas program to create and display array using pandas model

```
import pandas as pd

import numpy as np


def create_and_display_array():

    # Create a NumPy array

    data = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])


    # Create a DataFrame from the NumPy array

    df = pd.DataFrame(data, columns=['Column1', 'Column2', 'Column3'])


    # Display the DataFrame

    print("DataFrame created from NumPy array:")

    print(df)


def main():

    create_and_display_array()


if __name__ == "__main__":

    main()
```

OUTPUT :

```
DataFrame created from NumPy array:
   Column1  Column2  Column3
0        1        2        3
1        4        5        6
2        7        8        9
```

Ques 38: wap to convert pandas series to python list

```
import pandas as pd


def convert_series_to_list():
    # Create a Pandas Series
    data = pd.Series([10, 20, 30, 40, 50])

    # Convert the Pandas Series to a Python list
    data_list = data.tolist()

    # Display the original Series and the converted list
    print("Original Pandas Series:")
    print(data)
    print("\nConverted Python list:")
    print(data_list)


def main():
    convert_series_to_list()


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

OUTPUT :

```
Original    Pandas Series:
0              10
1              20
2              30
3              40
4              50
dtype: int64

Converted Python list:      [10, 20, 30, 40, 50]
```

Ques 39: wap to add,sub,mul and div of 2 series

```
import pandas as pd

def perform_operations(series1, series2):
    # Addition
    addition = series1 + series2

    # Subtraction
    subtraction = series1 - series2

    # Multiplication
    multiplication = series1 * series2

    # Division
    division = series1 / series2

    # Display the results
    print("Series 1:")
    print(series1)
    print("\nSeries 2:")
    print(series2)

    print("\nAddition of Series:")
    print(addition)
```

```
    print("\nSubtraction of Series:")
    print(subtraction)

    print("\nMultiplication of Series:")
    print(multiplication)

    print("\nDivision of Series:")
    print(division)

def main():
    # Create two Pandas Series
    series1 = pd.Series([10, 20, 30, 40, 50])
    series2 = pd.Series([5, 4, 3, 2, 1])

    # Perform operations on the Series
    perform_operations(series1, series2)

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

OUTPUT :

```
Series 1:
0   10
1   20
2   30
3   40
4   50
dtype: int64

Series 2:
0   5
1   4
2   3
3   2
```

```
4    1
dtype: int64

Addition of Series:
0    15
1    24
2    33
3    42
4    51
dtype: int64

Subtraction of Series:
0     5
1    16
2    27
3    38
4    49
dtype: int64

Multiplication of Series:
0    50
1    80
2    90
3    80
4    50
dtype: int64

Division of Series:
0     2.0
1     5.0
2    10.0
3    20.0
4    50.0
dtype: float64
```

Ques 40: program to convert numpy array to pandas series

```
import numpy as np
import pandas as pd

def convert_array_to_series(array):
    # Convert the NumPy array to a Pandas Series
    series = pd.Series(array)

    # Display the original array and the converted series
    print("Original NumPy array:")
    print(array)
    print("\nConverted Pandas Series:")
    print(series)

def main():
    # Create a NumPy array
    array = np.array([10, 20, 30, 40, 50])

    # Call the function to convert the array to a series and display the result
    convert_array_to_series(array)

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

OUTPUT :

```
Original NumPy array:
[10 20 30 40 50]
```

Converted Pandas Series:
0 10
1 20
2 30
3 40
4 50
dtype: int32

Ques 41: wap to create a display a dataframe from dic which has index label

```
import pandas as pd

# Example dictionary
data = {
   'Name': ['Alice', 'Bob', 'Charlie'],
   'Age': [25, 30, 35],
   'City': ['New York', 'Los Angeles', 'Chicago']
}

# Specifying index labels
index_labels = ['Person1', 'Person2', 'Person3']

# Creating the DataFrame
df = pd.DataFrame(data, index=index_labels)

# Displaying the DataFrame
print(df)
```

Output:

```
       Name  Age         City
Person1    Alice   25     New York
Person2      Bob   30  Los Angeles
Person3  Charlie   35      Chicago
```

Ques 42: Using the dirtydata csv fix the bad data set(bad data could be : empty cell ,data in wrong format ,duplicate data and wrong data)

```
import pandas as pd


# Load the dataset

df = pd.read_csv('dirtydata.csv')

# Display the initial DataFrame

print("Initial DataFrame:")

print(df)

df.dropna(inplace=True)

# df.fillna({

#     'column_name': 'fill_value',

#     'another_column_name': 'another_fill_value'

# }, inplace=True)

df['date_column'] = pd.to_datetime(df['date_column'], errors='coerce')

df['numeric_column'] = pd.to_numeric(df['numeric_column'], errors='coerce')

df.drop_duplicates(inplace=True)

df['age'] = df['age'].apply(lambda x: df['age'].mean() if x < 0 else x)

df.loc[df['score'] > 100, 'score'] = 100  # assuming scores should be between 0 and 100

print("Cleaned DataFrame:")

print(df)

df.to_csv('cleaneddata.csv', index=False)
```

Ques 43: wap to fix the wrong data using pandas (file used for the operation is dirtydata.csv)

```
import pandas as pd

# Load the dataset

df = pd.read_csv('dirtydata.csv')

# Display the initial DataFrame

print("Initial DataFrame:")

print(df)

# Step 1: Handle empty cells

# Option 1: Remove rows with any empty cells

df.dropna(inplace=True)

# Option 2: Fill empty cells with a specific value (e.g., fill numeric columns with 0 and string columns with 'Unknown')

# df.fillna({

#     'column_name': 'fill_value',

#     'another_column_name': 'another_fill_value'

# }, inplace=True)

# Step 2: Correct data formats

# Convert columns to appropriate data types

# Example: Convert a date column to datetime format

if 'date_column' in df.columns:

    df['date_column'] = pd.to_datetime(df['date_column'], errors='coerce')

# Example: Convert numeric columns to float or int

if 'numeric_column' in df.columns:

    df['numeric_column'] = pd.to_numeric(df['numeric_column'], errors='coerce')

# Step 3: Remove duplicate rows

df.drop_duplicates(inplace=True)

# Step 4: Correct incorrect data

# Example: Replace invalid values in a specific column
```

```
# Replace negative ages with the column's mean age (assuming age should be positive)
if 'age' in df.columns:
    df['age'] = df['age'].apply(lambda x: df['age'].mean() if x < 0 else x)
# Example: Replace values based on condition
# Replace outliers or specific incorrect values
if 'score' in df.columns:
    df.loc[df['score'] > 100, 'score'] = 100  # assuming scores should be between 0 and 100
# Additional example: Correct categorical data
if 'gender' in df.columns:
    df['gender'] = df['gender'].str.capitalize()  # standardize gender values to have capitalized format
# Display the cleaned DataFrame
print("Cleaned DataFrame:")
print(df)
# Save the cleaned DataFrame to a new CSV file
df.to_csv('cleaneddata.csv', index=False)
```

Ques 44:using pandas remove the duplicate data (file used for the operation is dirtydata.csv)

```
import pandas as pd


# Load the dataset
df = pd.read_csv('dirtydata.csv')
# Display the initial DataFrame
print("Initial DataFrame:")
print(df)
# Remove duplicate rows
df.drop_duplicates(inplace=True)
# Display the cleaned DataFrame
print("DataFrame after removing duplicates:")
print(df)
# Save the cleaned DataFrame to a new CSV file
df.to_csv('cleaneddata.csv', index=False)
```

ques 45: wap to find the sum of all elements in a list

```
# Define a list of numbers

numbers = [1, 2, 3, 4, 5]

# Calculate the sum of all elements in the list

total_sum = sum(numbers)

# Print the result

print("The sum of all elements in the list is:", total_sum)
```

Ques 46: wap to remove the duplicate from a list

```
# Define a list with duplicate elements

numbers = [1, 2, 3, 4, 5, 2, 3, 1]

# Remove duplicates by converting the list to a set

unique_numbers = list(set(numbers))

# Print the result

print("List after removing duplicates (order not preserved):", unique_numbers)
```

Ques 47: create a program to reverse a li8st without using build in function

```
# Define a list with some elements
numbers = [1, 2, 3, 4, 5]


# Initialize an empty list to store the reversed elements
reversed_numbers = []
# Iterate over the original list in reverse order and append elements to the new list
for i in range(len(numbers) - 1, -1, -1):
   reversed_numbers.append(numbers[i])
# Print the result
print("Reversed list:", reversed_numbers)
```

Ques 48: wap to find the max and min elements in the list

```
# Define a list with some elements
numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
# Initialize variables to store the maximum and minimum values
# Set them to the first element of the list initially
if len(numbers) == 0:
    raise ValueError("The list is empty, cannot find maximum and minimum values.")
max_element = numbers[0]
min_element = numbers[0]
# Iterate through the list to find the maximum and minimum values
for num in numbers:
    if num > max_element:
        max_element = num
    if num < min_element:
        min_element = num
# Print the results
print("The maximum element in the list is:", max_element)
print("The minimum element in the list is:", min_element)
```

Ques 49: implement a program to sort the list of integer in asc order without using build in function

```
# Define a list of integers

numbers = [64, 34, 25, 12, 22, 11, 90]

# Implementing Bubble Sort

def bubble_sort(arr):

    n = len(arr)

    # Traverse through all array elements

    for i in range(n):

        # Last i elements are already in place

        for j in range(0, n-i-1):

            # Traverse the array from 0 to n-i-1

            # Swap if the element found is greater

            # than the next element

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

                arr[j], arr[j+1] = arr[j+1], arr[j]


# Call the bubble_sort function

bubble_sort(numbers)

# Print the sorted list

print("
```

Ques 50:wap a python program to find the length of the tuple

```
# Define a tuple with some elements

my_tuple = (10, 20, 30, 40, 50)

# Find the length of the tuple using the len() function

length = len(my_tuple)


# Print the result

print("The length of the tuple is:", length)
```

Ques 51: implement a function to concatenate the two tuples

```
def concatenate_tuples(tuple1, tuple2):
    # Create a new tuple containing elements from both tuples
    concatenated_tuple = tuple1 + tuple2
    return concatenated_tuple
# Example tuples
tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)
# Concatenate the tuples using the function
result_tuple = concatenate_tuples(tuple1, tuple2)
# Print the result
print("Concatenated tuple:", result_tuple)
```

Ques 52: wap to find the index of an element in the tuple

```
def find_index(tuple, element):
    # Iterate through the tuple elements and find the index of the element
    for i in range(len(tuple)):
        if tuple[i] == element:
            return i
    # If element not found, return -1
    return -1
# Example tuple
my_tuple = (10, 20, 30, 40, 50)


# Element to find
element_to_find = 30
# Find the index of the element using the find_index function
index = find_index(my_tuple, element_to_find)
# Print the result
if index != -1:
    print("Index of", element_to_find, "in the tuple is:", index)
else:
    print("Element", element_to_find, "not found in the tuple.")
```

Ques 53: create a program to count the occurrence of an element in the tuple

```
def count_occurrence(tuple, element):
    # Initialize a counter to store the occurrence count
    count = 0
    # Iterate through the tuple elements and count the occurrences of the element
    for item in tuple:
        if item == element:
            count += 1
    return count
# Example tuple
my_tuple = (1, 2, 3, 1, 4, 1, 5)
# Element to count
element_to_count = 1
# Count the occurrence of the element using the count_occurrence function
occurrence_count = count_occurrence(my_tuple, element_to_count)
# Print the result
print("The occurrence count of", element_to_count, "in the tuple is:", occurrence_count)
```

Ques 54: write a function to check if two tuples have any element in common

```
def have_common_element(tuple1, tuple2):
    # Iterate through elements of the first tuple
    for element in tuple1:
        # Check if the element is present in the second tuple
        if element in tuple2:
            return True
    # If no common element found, return False
    return False
# Example tuples
tuple1 = (1, 2, 3, 4, 5)
tuple2 = (6, 7, 8, 9, 10)
# Check if the tuples have any common element using the have_common_element function
if have_common_element(tuple1, tuple2):
    print("The tuples have at least one common element.")
else:
    print("The tuples do not have any common element.")
```

Ques 55: wap to illiterate over a dic and print key values pair

```
# Define a dictionary
my_dict = {'name': 'Alice', 'age': 30, 'city': 'New York'}
# Iterate over the dictionary and print key-value pairs
for key in my_dict:
    print(key, ":", my_dict[key])
```

Ques 56: implement a function to merge two dic

```
def merge_dicts(dict1, dict2):
    merged_dict = dict1.copy()  # Make a copy of the first dictionary
    merged_dict.update(dict2)   # Update the copy with the second dictionary
    return merged_dict
# Example dictionaries
dict1 = {'a': 1, 'b': 2}
dict2 = {'b': 3, 'c': 4}
# Merge the dictionaries using the merge_dicts function
merged_dict = merge_dicts(dict1, dict2)
# Print the merged dictionary
print("Merged dictionary:", merged_dict)
```

Ques 57: wap to find the keys corresponding to the max and min value in the dic

```
def find_max_min_keys(dictionary):
    # Check if the dictionary is empty
    if not dictionary:
        return None, None
    # Initialize variables to store max and min values and their corresponding keys
    max_key, min_key = None, None
    max_value = float('-inf')  # Initialize to negative infinity
    min_value = float('inf')   # Initialize to positive infinity
    # Iterate over the dictionary items
    for key, value in dictionary.items():
        # Update max value and corresponding key
        if value > max_value:
            max_value = value
            max_key = key
        # Update min value and corresponding key
        if value < min_value:
            min_value = value
            min_key = key
    return max_key, min_key
# Example dictionary
my_dict = {'a': 10, 'b': 20, 'c': 5, 'd': 15}
# Find keys corresponding to max and min values using the function
max_key, min_key = find_max_min_keys(my_dict)
# Print the result
print("Key corresponding to the maximum value:", max_key)
print("Key corresponding to the minimum value:", min_key)
```

Ques 58: create a function to check if a key exists in the dic

```
def key_exists(dictionary, key):
    # Check if the key exists in the dictionary
    return key in dictionary
# Example dictionary
my_dict = {'a': 10, 'b': 20, 'c': 30}
# Key to check
key_to_check = 'b'
# Check if the key exists using the key_exists function
if key_exists(my_dict, key_to_check):
    print("Key", key_to_check, "exists in the dictionary.")
else:
    print("Key", key_to_check, "does not exist in the dictionary.")
```

Ques 59: wap to sort a dic by its value in asc order

```
def sort_dict_by_value_asc(dictionary):
    # Use sorted() function with a lambda function as key to sort dictionary by value in ascending order
    sorted_dict = dict(sorted(dictionary.items(), key=lambda item: item[1]))
    return sorted_dict

# Example dictionary
my_dict = {'a': 10, 'b': 5, 'c': 20, 'd': 15}

# Sort the dictionary by value in ascending order using the sort_dict_by_value_asc function
sorted_dict = sort_dict_by_value_asc(my_dict)

# Print the sorted dictionary
print("Sorted dictionary by value in ascending order:", sorted_dict)
```

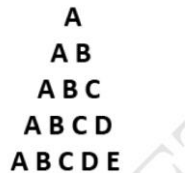
Ques 60: wap to create a set and perform basic set operation (union, intersection and difference)

```
# Create two sets

set1 = {1, 2, 3, 4, 5}

set2 = {4, 5, 6, 7, 8}

# Print the original sets

print("Set 1:", set1)

print("Set 2:", set2)

# Perform union of the sets

union_set = set1.union(set2)

print("Union of set 1 and set 2:", union_set)

# Perform intersection of the sets

intersection_set = set1.intersection(set2)

print("Intersection of set 1 and set 2:", intersection_set)

# Perform difference of the sets (set1 - set2)

difference_set1 = set1.difference(set2)

print("Difference of set 1 and set 2:", difference_set1)

# Perform difference of the sets (set2 - set1)

difference_set2 = set2.difference(set1)

print("Difference of set 2 and set 1:", difference_set2)
```

Ques 62: wap to check if two sets have any elements in common

```
def have_common_elements(set1, set2):
    # Check if the intersection of the sets is not empty
    return len(set1.intersection(set2)) > 0
# Example sets
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
# Check if the sets have any common elements using the have_common_elements function
if have_common_elements(set1, set2):
    print("The sets have common elements”)
```

Ques 63: draw pattern:




```
n=int (input("Enter the number of rows:"))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(1,i+1):
        print(chr(64+j),end=" ")
    print()
```

Ques 64: draw pattern:

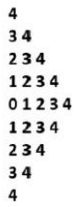
```
E E E E E E E E E
 D D D D D D D
  C C C C C
   B B B
    A
```

```
num=int(input("Enter the number of rows: "))
for i in range(1,num+1):
    print(" "*(i-1),end="")
    for j in range(1,num+2-i):
        print(chr(65+num-i),end=" ")
    for k in range(2,num+2-i):
        print(chr(65+num-i),end=" ")
    print()
```

Ques 65: draw pattern :


```
num=int(input("Enter a  number:"))
for i in range(1,num+1):
    print(" "*(num-i),end="")
    for j in range(1,i+1):
        print(num-j,end=" ")
    print()
for k in range(1,num):
    print(" "*k,end="")
    for l in range(1,num+1-k):
        print(num-1,end=" ")
    print()
```

Ques 66: draw pattern :




```
num=int(input("Enter a number:"))
for i in range(1,num+1):
   for j in range(1,i+1):
      print(num-i+j-1,end="")
   print()
for a in range(1,num+1):
   for k in range(0,num-a):
      print(k+a,end="")
   print()
```

Ques 67: draw pattern :


```
num=int(input("Enter a number:"))
for i in range(1,num+1):
    print(" "*(num-i),end="")
    for j in range(0,i):
        print(chr(65+num+j-i),end=" ")
    print()
for k in range(1,num):
    print(" "*k,end="")
    for l in range(0,num-k):
        print(chr(65+k+1),end=" ")
    print()
```

Question 68==print the pattern

```
num=int(input("Enter a number:"))
for i in range(1,num+1):
    print(" "*(num-i),end="")
    for j in range(1,i+1):
        print("*",end=" ")
    print(" "*(num-i),end="")
    for k in range(1,i+1):
        print("*",end=" ")
    print()
```