

**Lab Practical File**

**Python Programming Lab**

**PBC 401**

**Submitted To: - Submitted By: -**

**Dr. Janmejay Pant Ashish Almiya (***Assistant Professor***)** **BCA (4th Sem.) Graphic Era Hill University Roll no. 2271045**

**Bhimtal Campus**

# Ques 1 : WAP to purposefully raise the indentation error and correct it ?

def greet(name):

print("Hello, " + name) greet("Alice")

output:

# Cell In[1], line 2 print("Hello, " + name)

**^**

**IndentationError:** expected an indented block after function definition on line 1

# Ques 2: WAP to demonstrate the use of arithmetic operation in python ?

a = 10

b = 5

add\_result = a + b

print("Addition: {} + {} = {}".format(a, b, add\_result)) sub\_result = a - b

print("Subtraction: {} - {} = {}".format(a, b, sub\_result)) mul\_result = a \* b

print("Multiplication: {} \* {} = {}".format(a, b, mul\_result)) div\_result = a / b

print("Division: {} / {} = {}".format(a, b, div\_result)) mod\_result = a % b

print("Modulus: {} % {} = {}".format(a, b, mod\_result)) exp\_result = a \*\* b

print("Exponentiation: {} \*\* {} = {}".format(a, b, exp\_result))

output

Addition: 10 + 5 = 15

Subtraction: 10 - 5 = 5

Multiplication: 10 \* 5 = 50

Division: 10 / 5 = 2.0

Modulus: 10 % 5 = 0

Exponentiation: 10 \*\* 5 = 100000

# Ques 3: WAP that takes command line argument as input and print the addition of two numbers as output.

import sys def main():

if len(sys.argv) != 3:

print("Usage: python add\_numbers.py <num1> <num2>") return

try:

num1 = int(sys.argv[1]) num2 = int(sys.argv[2]) result = num1 + num2

print("The addition of {} and {} is: {}".format(num1, num2, result)) except ValueError:

print("Please provide valid integers as input.")

if name == " main ": main()

OUTPUT :

python add\_numbers.py 5 10

The addition of 5 and 10 is:

# Ques 4: WAP to compute distance between two points taking input from user

**D=√(x2 - x1)^2 + (y2 - y1)^2**

import math

def compute\_distance(x1, y1, x2, y2):

# Calculate the distance using the distance formula distance = math.sqrt((x2 - x1)\*\*2 + (y2 - y1)\*\*2) return distance

def main():

try:

# Taking input from the user

x1 = float(input("Enter the x-coordinate of the first point: ")) y1 = float(input("Enter the y-coordinate of the first point: "))

x2 = float(input("Enter the x-coordinate of the second point: ")) y2 = float(input("Enter the y-coordinate of the second point: ")) distance = compute\_distance(x1, y1, x2, y2)

print("The distance between the points ({}, {}) and ({}, {}) is: {:.2f}".format(x1, y1, x2, y2, distance))

except ValueError:

print("Please enter valid numeric values.")

if name == " main ": main()

OUTPUT :

Enter the x-coordinate of the first point: 3 Enter the y-coordinate of the first point: 4 Enter the x-coordinate of the second point: 7

Enter the y-coordinate of the second point: 1

The distance between the points (3.0, 4.0) and (7.0, 1.0) is: 5.00

# Ques 5: create a string “hello world ” and perform the following operation :

**A: print the complete string**

# B: print the first character of the string C: print the character string from 3 to 5 D: print string starting from 3rd character E: print the string 20 times

**F: create a new string and concatenate with previous string**

# Define the initial string initial\_string = "hello world"

# A: Print the complete string

print("A: The complete string is: '{}'".format(initial\_string))

# B: Print the first character of the string

print("B: The first character of the string is: '{}'".format(initial\_string[0]))

# C: Print the character string from 3 to 5

# Note: Indexing in Python starts from 0, so characters from index 2 to 4 are the 3rd to 5th characters.

print("C: The characters from 3 to 5 are: '{}'".format(initial\_string[2:5]))

# D: Print the string starting from the 3rd character

print("D: The string starting from the 3rd character is: '{}'".format(initial\_string[2:]))

# E: Print the string 20 times

print("E: The string printed 20 times is: '{}'".format(initial\_string \* 20))

# F: Create a new string and concatenate with the previous string new\_string = " Have a great day!"

concatenated\_string = initial\_string + new\_string

print("F: The concatenated string is: '{}'".format(concatenated\_string))

OUTPUT :

A: The complete string is: 'hello world' B: The first character of the string is: 'h' C: The characters from 3 to 5 are: 'llo'

D: The string starting from the 3rd character is: 'llo world'

E: The string printed 20 times is: 'hello worldhello worldhello worldhello worldhello worldhello worldh ello worldhello worldhello worldhello worldhello worldhello worldhello worldhello worldhello worldh ello worldhello worldhello worldhello worldhello world'

F: The concatenated string is: 'hello world Have a great day!'

# Ques 6: Create a List containing the following items and perform the following operations-

**Student =[ 556,”Mothi”,84,96,84,75,84]**

# Access 0th element

1. **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

# Define the initial list

Student = [556, "Mothi", 84, 96, 84, 75, 84]

# 1. Access 0th element

print("1. The 0th element is: {}".format(Student[0]))

# 2. Access 0th to 1st elements

print("2. The 0th to 1st elements are: {}".format(Student[0:2]))

# 3. Access 2nd to end of list elements

print("3. The 2nd to end of list elements are: {}".format(Student[2:]))

# 4. Access starting to 2nd elements

print("4. The starting to 2nd elements are: {}".format(Student[:2]))

# 5. Access starting to ending elements

print("5. The starting to ending elements are: {}".format(Student[:]))

# 6. Access last index value

print("6. The last index value is: {}".format(Student[-1]))

# 7. Access elements in reverse order

print("7. The elements in reverse order are: {}".format(Student[::-1]))

OUTPUT :

1. The 0th element is: 556
2. The 0th to 1st elements are: [556, 'Mothi']
3. The 2nd to end of list elements are: [84, 96, 84, 75, 84]
4. The starting to 2nd elements are: [556, 'Mothi']
5. The starting to ending elements are: [556, 'Mothi', 84, 96, 84, 75, 84]
6. The last index value is: 84
7. The elements in reverse order are: [84, 75, 84, 96, 84, 'Mothi', 556]

# Ques 7: Create DataFrame

import pandas as pd

# Define the data as a dictionary data = {

'Student\_ID': [556, 557, 558],

'Name': ['Mothi', 'Alice', 'Bob'], 'Math\_Score': [84, 90, 78],

'Science\_Score': [96, 85, 88],

'English\_Score': [84, 79, 92]

}

# Create the DataFrame df = pd.DataFrame(data)

# Print the DataFrame print("DataFrame:") print(df)

OUTPUT :

DataFrame:

Student\_ID Name Math\_Score Science\_Score English\_Score

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 556 Mothi | 84 | 96 | 84 |
| 1 | 557 Alice | 90 | 85 | 79 |
| 2 | 558 Bob | 78 | 88 | 92 |

# Ques 8: write a program to check whether the given no is even or odd

def check\_even\_odd(number):

if number % 2 == 0:

return "even" else:

return "odd"

def main():

try:

# Take input from the user

number = int(input("Enter a number: "))

# Check whether the number is even or odd result = check\_even\_odd(number)

# Print the result

print("The number {} is {}.".format(number, result)) except ValueError:

print("Please enter a valid integer.")

if name == " main ": main()

OUTPUT :

Enter a number: 12 The number 12 is even.

# Ques 9 :Write a program to get the marks of 5 subjects and calculate the average marks scored and calculate the grade based on the average marks

**If**

# Avg 91 to 100 (A grade)

**71 to 80 (C grade )**

# 61 to 60 (D grade)

**51 to 60 (E grade)**

# >50 (fail)

def calculate\_grade(average):

if average >= 91 and average <= 100: return 'A'

elif average >= 81 and average <= 90: return 'B'

elif average >= 71 and average <= 80: return 'C'

elif average >= 61 and average <= 70: return 'D'

elif average >= 51 and average <= 60: return 'E'

else:

return 'Fail'

def main():

try:

# Take input for the marks of 5 subjects marks = []

for i in range(1, 6):

mark = float(input(f"Enter the marks for subject {i}: ")) marks.append(mark)

# Calculate the average marks average = sum(marks) / len(marks)

# Calculate the grade based on the average marks grade = calculate\_grade(average)

# Print the results

print(f"\nAverage Marks: {average:.2f}") print(f"Grade: {grade}")

except ValueError:

print("Please enter valid numeric values for the marks.")

if name == " main ":

main()

OUTPUT :

Enter the marks for subject 1: 77 Enter the marks for subject 2: 93 Enter the marks for subject 3: 69 Enter the marks for subject 4: 90 Enter the marks for subject 5: 59

Average Marks: 77.60 Grade: C

# Ques 10: Write a program to implement the simple calculator that performs basic mathematical operation it takes 3 inputs from the user at first the user is asked to enter 2

**numbers after that the user is asked to enter the operator based on which arithmetic operation is performed**

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 the first number num1 = float(input("Enter the first number: "))

# Take input from the user for the second number num2 = float(input("Enter the second number: "))

# Take input from the user for the operator operator = input("Enter the operator (+, -, \*, /): ")

# Perform the calculation based on the operator if operator == '+':

result = add(num1, num2) elif operator == '-':

result = subtract(num1, num2) elif operator == '\*':

result = multiply(num1, num2) elif operator == '/':

result = divide(num1, num2) else:

result = "Invalid operator entered."

# Print the result print("Result:", result)

except ValueError:

print("Please enter valid numeric values for the numbers.") except Exception as e:

print("An error occurred:", str(e))

if name == " main ": main()

OUTPUT :

Enter the first number: 10 Enter the second number: 5 Enter the operator (+, -, \*, /): + Result: 15.0

Ques 11: Write a program to check whether the character enter by the user is in upper case or in lower case

def check\_case(char):

if char.islower():

return "Lowercase" elif char.isupper():

return "Uppercase" else:

return "Not an alphabet"

def main():

try:

# Take input from the user

char = input("Enter a character: ")

# Check the case of the character case = check\_case(char)

# Print the result

print("The character '{}' is in {}.".format(char, case)) except Exception as e:

print("An error occurred:", str(e))

if name == " main ": main()

OUTPUT :

Enter a character: a

The character 'a' is in Lowercase.

Ques 12: Write a program that ask user to enter the number and if the entered number is greater that or equal to 100 check the number is even or odd if number is less that 100 it will not check the number is even or not it will only print the message that the number is less 100

def check\_even\_odd(number):

if number % 2 == 0:

return "even" else:

return "odd"

def main():

try:

# Take input from the user

number = int(input("Enter a number: "))

# Check if the number is greater than or equal to 100 if number >= 100:

# Check if the number is even or odd result = check\_even\_odd(number)

print("The number {} is greater than or equal to 100 and it is {}.".format(number, result))

else:

print("The number {} is less than 100.".format(number)) except ValueError:

print("Please enter a valid integer.")

if name == " main ":

main()

OUTPUT :

Enter a number: 103

The number 103 is greater than or equal to 100 and it is odd.

Ques 13: Write a program to check whether the given year is leap or not

def is\_leap\_year(year):

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0): return True

else:

return False

def main():

try:

# Take input from the user

year = int(input("Enter a year: "))

# Check if the year is a leap year if is\_leap\_year(year):

print("{} is a leap year.".format(year)) else:

print("{} is not a leap year.".format(year)) except ValueError:

print("Please enter a valid integer.")

if name == " main ":

main()

OUTPUT :

Enter a year: 2024 2024 is a leap year.

Ques 14: write a program to print even numbers upto n using while loop

def print\_even\_numbers(n):

i = 2

while i <= n:

print(i, end=" ") i += 2

def main():

try:

# Take input from the user for the value of n n = int(input("Enter the value of n: "))

# Print even numbers up to n

print("Even numbers up to {}:".format(n)) print\_even\_numbers(n)

except ValueError:

print("Please enter a valid integer.")

if name == " main ":

main()

OUTPUT :

Enter the value of n: 100 Even numbers up to 100:

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 52 54 56 58 60 62 64 66 68 70

72 74 76 78 80 82 84 86 88 90 92 94 96 98 100

Ques 15: wap to find the sum of the two matrixes using for loop

def add\_matrices(matrix1, matrix2):

if len(matrix1) != len(matrix2) or len(matrix1[0]) != len(matrix2[0]): return "Matrices are not of the same size"

result = []

for i in range(len(matrix1)):

row = []

for j in range(len(matrix1[0])): row.append(matrix1[i][j] + matrix2[i][j])

result.append(row)

return result

def main():

try:

# Take input for the dimensions of the matrices rows = int(input("Enter the number of rows: ")) cols = int(input("Enter the number of columns: "))

# Take input for the elements of the first matrix print("Enter elements of the first matrix:")

matrix1 = [[int(input("Enter element [{},{}]: ".format(i, j))) for j in range(cols)] for i in range(rows)]

# Take input for the elements of the second matrix print("Enter elements of the second matrix:")

matrix2 = [[int(input("Enter element [{},{}]: ".format(i, j))) for j in range(cols)] for i in range(rows)]

# Calculate the sum of the matrices result = add\_matrices(matrix1, matrix2)

# Print the result

if isinstance(result, str):

print(result) else:

print("Sum of the matrices:") for row in result:

print(row) except ValueError:

print("Please enter valid integer values for the matrix elements.")

if name == " main ": main()

OUTPUT :

Enter the number of rows: 2 Enter the number of columns: 2

Enter elements of the first matrix:

Enter element [0,0]: 1

Enter element [0,1]: 2

Enter element [1,0]: 3

Enter element [1,1]: 4

Enter elements of the second matrix:

Enter element [0,0]: 5

Enter element [0,1]: 6

Enter element [1,0]: 7

Enter element [1,1]: 8 Sum of the matrices:

[6, 8]

[10, 12]

Ques 16: wap to find the sum of n natural numbers using while loop and for loop and should be entered by the user

def sum\_of\_natural\_numbers\_while(n):

sum = 0

i = 1

while i <= n:

sum += i i += 1

return sum

def main\_while():

try:

n = int(input("Enter the value of n: ")) if n < 1:

print("Please enter a positive integer.") return

result = sum\_of\_natural\_numbers\_while(n)

print("Sum of the first {} natural numbers using while loop: {}".format(n, result)) except ValueError:

print("Please enter a valid integer.")

if name == " main ":

main\_while()

OUTPUT :

Enter the value of n: 10

Sum of the first 10 natural numbers using while loop: 55

def sum\_of\_natural\_numbers\_for(n):

sum = 0

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

sum += i return sum

def main\_for():

try:

n = int(input("Enter the value of n: ")) if n < 1:

print("Please enter a positive integer.") return

result = sum\_of\_natural\_numbers\_for(n)

print("Sum of the first {} natural numbers using for loop: {}".format(n, result)) except ValueError:

print("Please enter a valid integer.")

if name == " main ": main\_for()

OUTPUT :

Enter the value of n: 10

Sum of the first 10 natural numbers using for loop: 55

Ques 17: print the pattern

1

22

333

4444

55555

def print\_pattern(rows):

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

print(str(i) \* i)

def main():

try:

rows = int(input("Enter the number of rows: ")) print("Pattern:")

print\_pattern(rows) except ValueError:

print("Please enter a valid integer.")

if name == " main ": main()

OUTPUT :

Enter the number of rows: 5 Pattern:

1

22

333

4444

55555

Ques 18: wap to calculate the power of the number using the anonymous function

# Define an anonymous function to calculate the power of a number power = lambda base, exponent: base \*\* exponent

def main():

try:

# Take input from the user for the base and exponent base = float(input("Enter the base number: ")) exponent = float(input("Enter the exponent: "))

# Calculate the power using the anonymous function result = power(base, exponent)

# Print the result print("Result:", result)

except ValueError:

print("Please enter valid numeric values for the base and exponent.")

if name == " main ": main()

OUTPUT :

Enter the base number: 2 Enter the exponent: 3 Result: 8.0

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

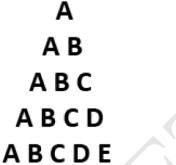
# Example sets set1 = {1, 2, 3, 4, 5}

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:**

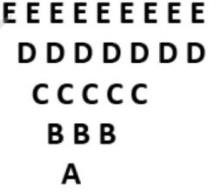


n=int (input("Enter the number of rows:")) for i in range(1,n+1):

print(" "\*(n-i),end="") for j in range(1,i+1):

print(chr(64+j),end=" ") print()

**Ques 64: draw pattern**:



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):

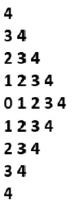
print(num-j,end=" ") print()

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()

**Ques 68: 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("\*",end=" ") print(" "\*(num-i),end="") for k in range(1,i+1):

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