lOMoARcPSD|52710018

**DEPARTMENT**

**OF**

**MASTER OF COMPUTER APPLICATIONS**

**LAB MANUAL R23**

# PYTHON PROGRAMMING (SKILL ENHANCEMENT PROGRAM)

# I MCA – II SEMESTER

# 2024-25

**.**

**PREFACE OF THE LAB MANUAL**

|  |  |
| --- | --- |
| **Academic Year** | **: 2024-25** |
| **Department for which Subject is taken** | **: MASTER OF COMPUTER APPLICATIONS** |
| **Course Name** | **: I - MCA** |
| **Student’s Batch** | **: 2024-2026** |
| **Name of the Subject** | **: PYTHON PROGRMMING** |
| **Regulation** | **: R23** |
| **Course Code** | **:23F01003** |
| **Faculty in-charge** | **: HIMAMBASHA SHAIK** |

VISION & MISSION OF COLLEGE

**Vision:**

* To be a recognized premier institution in engineering education, research, and application of knowledge to benefit society.

**Mission:**

* Impart technical knowledge at all levels to transform the engineers capable to address the challenges.
* Inculcate competencies for all-round development to meet industrial, technical, and leadership skills.
* Provide learner-centric ambiance with collaborations.
* Promote creativity, knowledge development, ethical values, and interpersonal skills.

Course Objectives:

The main objectives of the course are to

* Introduce core programming concepts of Python programming language.
* Demonstrate about Python data structures like Lists, Tuples, Sets and dictionaries
* Implement Functions, Modules and Regular Expressions in Python Programming and to create Practical and contemporary applications using these

**UNIT 1**

1. Write a program to find the largest element among three Numbers.
2. Write a Program to display all prime numbers within an interval
3. Write a program to swap two numbers without using a temporary variable.
4. Demonstrate the following Operators in Python with suitable examples.
5. Arithmetic Operators
6. Relational Operators
7. Assignment Operators
8. Logical Operators
9. Bit wise Operators
10. Ternary Operator
11. Membership Operators
12. Identity Operators
13. Write a program to add and multiply complex numbers
14. Write a program to print multiplication table of a given number.

**UNIT 2**

1. Write a program to define a function with multiple return values.
2. Write a program to define a function using default arguments.
3. Write a program to find the length of the string without using any library functions.
4. Write a program to check if the substring is present in a given string or not.
5. Write a program to perform the given operations on a list:

i. addition ii. Insertion iii. Slicing

1. Write a program to perform any 5 built-in functions by taking any list.

**UNIT-III**

1. Write a program to create tuples (name, age, address, college) for at least two members and concatenate the tuples and print the concatenated tuples.
2. Write a program to count the number of vowels in a string (No control flow allowed).
3. Write a program to check if a given key exists in a dictionary or not.
4. Write a program to add a new key-value pair to an existing dictionary.
5. Write a program to sum all the items in a given dictionary.

**UNIT IV:**

1. Write a program to sort words in a file and put them in another file. The output file should have only lower-case words, so any upper-case words from source must be lowered.
2. Python program to print each line of a file in reverse order.
3. Python program to compute the number of characters, words and lines in a file.
4. Write a program to create, display, append, insert and reverse the order of the items in the array.
5. Write a program to add, transpose and multiply two matrices.
6. Write a Python program to create a class that represents a shape. Include methods to calculate its area and perimeter. Implement subclasses for different shapes like circle, triangle, and square.

**UNIT V**

1. Python program to check whether a JSON string contains complex object or not.
2. Python Program to demonstrate NumPy arrays creation using array () function.
3. Python program to demonstrate use of ndim, shape, size, dtype.
4. Python program to demonstrate basic slicing, integer and Boolean indexing.
5. Python program to find min, max, sum, cumulative sum of array
6. Create a dictionary with at least five keys and each key represent value as a list where this list contains at least ten values and convert this dictionary as a pandas data frame and explore the data through the data frame as follows:
7. Apply head () function to the pandas data frame
8. Perform various data selection operations on Data Frame

7. Select any two columns from the above data frame, and observe the change in one attribute with respect to other attribute with scatter and plot operations in matplotlib

# UNIT -1

**1. Find the largest element among three numbers**

# Program to find the largest element among three numbers

# Function to find the largest of three numbers def find\_largest(num1, num2, num3):

if num1 >= num2 and num1 >= num3: return num1

elif num2 >= num1 and num2 >= num3: return num2

else:

return num3 # Taking input from user

num1 = float(input("Enter first number: ")) num2 = float(input("Enter second number: ")) num3 = float(input("Enter third number: "))

# Output

largest = find\_largest(num1, num2, num3)

print(f"The largest number among {num1}, {num2}, and {num3} is {largest}.")

**2. Display all prime numbers within an interval**

# Program to display all prime numbers within an interval

# Function to check if a number is prime def is\_prime(num): if num <= 1: return False for i in range(2, int(num \*\* 0.5) + 1): if num % i == 0: return False return True

# Function to display prime numbers within an interval def display\_primes(start, end):

primes = [] for num in range(start, end + 1): if is\_prime(num):

primes.append(num) return primes # Taking input from user start = int(input("Enter start of interval: ")) end = int(input("Enter end of interval: "))

# Output

primes = display\_primes(start, end)

print(f"Prime numbers between {start} and {end} are: {primes}")

**3. Swap two numbers without using a temporary variable**

# Program to swap two numbers without using a temporary variable

# Function to swap two numbers def swap\_numbers(a, b): a = a + b b = a - b a = a - b return a, b

# Taking input from user a = float(input("Enter first number: ")) b = float(input("Enter second number: "))

# Output a, b = swap\_numbers(a, b) print(f"After swapping: a = {a}, b = {b}")

**4. Demonstrate the following operators in Python with suitable examples**

**# i) Arithmetic Operators**

# Arithmetic Operators

a = float(input("Enter first number for arithmetic operations: ")) b = float(input("Enter second number for arithmetic operations: ")) print("Addition:", a + b) print("Subtraction:", a - b) print("Multiplication:", a \* b) print("Division:", a / b) print("Modulus:", a % b) print("Exponentiation:", a \*\* b) print("Floor Division:", a // b)

**# ii) Relational Operators**

# Relational Operators

a = float(input("Enter first number for relational operations: "))

b = float(input("Enter second number for relational operations: "))

print("a == b:", a == b)

print("a != b:", a != b)

print("a > b:", a > b)

print("a < b:", a < b)

print("a >= b:", a >= b)

print("a <= b:", a <= b)

**# iii) Assignment Operators**

# Assignment Operators

a = float(input("Enter a number for assignment operations: "))

b = float(input("Enter another number for assignment operations: ")) a += b

print("a += b:", a)

a -= b

print("a -= b:", a)

a \*= b

print("a \*= b:", a)

a /= b

print("a /= b:", a)

a %= b

print("a %= b:", a)

a \*\*= b

print("a \*\*= b:", a)

a //= b

print("a //= b:", a)

**# iv) Logical Operators**

# Logical Operators

a = bool(int(input("Enter 1 for True or 0 for False for logical operations (a): "))) b = bool(int(input("Enter 1 for True or 0 for False for logical operations (b): "))) print("a and b:", a and b)

print("a or b:", a or b)

print("not a:", not a)

**# v) Bitwise Operators**

# Bitwise Operators

a = int(input("Enter an integer for bitwise operations (a): "))

b = int(input("Enter another integer for bitwise operations (b): ")) print("a & b:", a & b)

print("a | b:", a | b)

print("a ^ b:", a ^ b)

print("~a:", ~a)

print("a << 2:", a << 2)

print("a >> 2:", a >> 2)

**# vi) Ternary Operator**

# Ternary Operator

a = float(input("Enter first number for ternary operation: "))

b = float(input("Enter second number for ternary operation: ")) result = "a is greater"

if a > b

else "b is greater"

print(result)

**# vii) Membership Operators**

# Membership Operators

a= input("Enter a list of numbers separated by space: ").split()

a = [int(i) for i in a]

b= int(input("Enter a number to check membership in the list:"))

c = int(input("Enter another number to check membership in the list: "))

print("b in a:", b in a)

print("c not in a:", c not in a)

**# viii)** **Identity Operators**

# Identity Operators a = [1, 2, 3] b = a c = [1, 2, 3]

print("a is b:", is b)

print("a is c:", a is c)

print("a == c:", a == c)

**5. Add and multiply complex numbers**

# Program to add and multiply complex numbers

# Taking input from user

real1 = float(input("Enter the real part of first complex number: "))

imag1 = float(input("Enter the imaginary part of first complex number: "))

real2 = float(input("Enter the real part of second complex number: "))

imag2 = float(input("Enter the imaginary part of second complex number: "))

c1 = complex(real1, imag1)

c2 = complex(real2, imag2)

# Addition add = c1 + c2 # Multiplication multiply = c1 \* c2 # Output

print(f"Addition of {c1} and {c2} is {add}") print(f"Multiplication of {c1} and {c2} is {multiply}")

**6. Print multiplication table of a given number**

# Program to print multiplication table of a given number

# Function to print multiplication table def multiplication\_table(n): for i in range(1, 11): print(f"{n} x {i} = {n \* i}") # Taking input from user

n = int(input("Enter a number to print its multiplication table: "))

# Output multiplication\_table(n) **UNIT – II**

1. Define a function with multiple return values

# Program to define a function with multiple return values

# Function that returns multiple values def calculate(a, b): sum\_ = a + b diff = a - b prod = a \* b

quot = a / b if b != 0 else None return sum\_, diff, prod, quot # Taking input from user a = float(input("Enter first number: ")) b = float(input("Enter second number: "))

# Output

sum\_, diff, prod, quot = calculate(a, b)

print(f"Sum: {sum\_}, Difference: {diff}, Product: {prod}, Quotient: {quot}")

2. Define a function using default arguments

# Program to define a function using default arguments

# Function with default arguments def greet(name, greeting="Hello"): return f"{greeting}, {name}!" # Taking input from user name = input("Enter your name: ") greeting = input("Enter a greeting (optional): ")

# Output if greeting: print(greet(name, greeting)) else:

print(greet(name))

3. Find the length of the string without using any library functions

# Program to find the length of the string without using any library functions

# Function to calculate length of string def string\_length(s):

length = 0 for char in s: length += 1 return length

# Taking input from user s = input("Enter a string: ")

# Output length = string\_length(s)

print(f"The length of the string '{s}' is {length}.")

4. Check if the substring is present in a given string or not

# Program to check if the substring is present in a given string or not

# Function to check if substring is present def is\_substring(main\_string, sub\_string):

return sub\_string in main\_string # Taking input from user

main\_string = input("Enter the main string: ") sub\_string = input("Enter the substring: ")

# Output if is\_substring(main\_string, sub\_string):

print(f"The substring '{sub\_string}' is present in '{main\_string}'.")

else:

print(f"The substring '{sub\_string}' is not present in '{main\_string}'.")

5. Perform operations on a list: addition, insertion, slicing

# Program to perform operations on a list # Function to perform operations on list def list\_operations(lst): # Addition lst.append(10) # Insertion lst.insert(1, 20) # Slicing sliced\_list = lst[1:4] return lst, sliced\_list # Taking input from user

lst = list(map(int, input("Enter a list of numbers separated by space: ").split())

# Output

updated\_list, sliced\_list = list\_operations(lst) print(f"Updated List: {updated\_list}") print(f"Sliced List: {sliced\_list}")

6. Perform any 5 built-in functions by taking any list

# Program to perform any 5 built-in functions on a list # Function to demonstrate built-in functions def built\_in\_functions(lst):

length = len(lst) max\_val = max(lst) min\_val = min(lst[)](https://www.studocu.com/in?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=ii-btech-i-semester-r23-python-programming-lab-manual)  sum\_val = sum(lst) sorted\_list = sorted(lst)

return length, max\_val, min\_val, sum\_val, sorted\_list

# Taking input from user

lst = list(map(int, input("Enter a list of numbers separated by space: ").split()))

# Output

length, max\_val, min\_val, sum\_val, sorted\_list = built\_in\_functions(lst) print(f"Length: {length}") print(f"Max Value: {max\_val}") print(f"Min Value: {min\_val}") print(f"Sum: {sum\_val}") print(f"Sorted List: {sorted\_list}")

# UNIT -III

1. **Create and concatenate tuples**

# Program to create and concatenate tuples # Creating tuples for two members

member1 = ("Alice", 21, "123 Maple Street", "XYZ College")

member2 = ("Bob", 22, "456 Oak Avenue", "ABC University")

# Concatenating tuples

concatenated\_tuple = member1 + member2

# Output the concatenated tuple

print("Concatenated Tuple:", concatenated\_tuple)

1. **Count the number of vowels in a string (No control flow allowed)**

# Program to count the number of vowels in a string (No control flow allowed)

# Function to count vowels

def count\_vowels(s):

return sum(s.count(vowel) for vowel in "aeiouAEIOU")

# Taking input from user s = input("Enter a string: ") # Output the number of vowels print("Number of vowels:", count\_vowels(s))

1. **Check if a given key exists in a dictionary or not**

# Program to check if a given key exists in a dictionary or not

# Function to check if key exists def key\_exists(d, key):

return key in d

# Taking input from user

d = {'name': 'Alice', 'age': 21, 'address': '123 Maple Street'} key = input("Enter a key to check: ") # Output if the key exists or not if key\_exists(d, key):

print(f"The key '{key}' exists in the dictionary.") else: print(f"The key '{key}' does not exist in the dictionary.")

**4. Add a new key-value pair to an existing dictionary**

# Program to add a new key-value pair to an existing dictionary

# Function to add key-value pair def add\_key\_value(d, key, value):

d[key] = value return d

# Taking input from user

d = {'name': 'Alice', 'age': 21, 'address': '123 Maple Street'} key = input("Enter a key to add: ") value = input("Enter a value to add: ") # Output the updated dictionary updated\_dict = add\_key\_value(d, key, value) print("Updated Dictionary:", updated\_dict)

**5. Sum all the items in a given dictionary**

# Program to sum all the items in a given dictionary # Function to sum all values in the dictionary def sum\_dict\_items(d): return sum(d.values())

# Taking input from user

d = {'item1': 100, 'item2': 200, 'item3': 300} # Output the sum of all items total\_sum = sum\_dict\_items(d) print("Sum of all items:", total\_sum)

**UNIT IV**

**1: Sort words in a file and put them in another file**

def sort\_words(input\_file, output\_file): with open(input\_file, 'r') as file: words = file.read().split()

# Convert words to lowercase and sort them words = sorted([word.lower() for word in words])

with open(output\_file, 'w') as file: file.write("\n".join(words)) # Example usage:

input\_file = 'input.txt' output\_file = 'output.txt' sort\_words(input\_file, output\_file) print(f"Words sorted and saved to '{output\_file}'")

**2: Print each line of a file in reverse order**

def reverse\_lines(input\_file): with open(input\_file, 'r') as file:

lines = file.readlines() for line in reversed(lines):

print(line.rstrip())

# Example usage:

input\_file = 'input.txt' reverse\_lines(input\_file) **3: Compute number of characters, words, and lines in a file**

def count\_chars\_words\_lines(input\_file): with open(input\_file, 'r') as file:

lines = file.readlines() num\_lines = len(lines)

num\_words = sum(len(line.split()) for line in lines) num\_chars = sum(len(line) for line in lines) print(f"Number of lines: {num\_lines}") print(f"Number of words: {num\_words}") print(f"Number of characters: {num\_chars}") # Example usage:

input\_file = 'input.txt' count\_chars\_words\_lines(input\_file)

**4: Operations on an array (create, display, append, insert, reverse)**

class ArrayOperations: def \_\_init\_\_(self): self.array = [] def create\_array(self, elements):

self.array = elements def display\_array(self): print("Array:", self.array) def append\_element(self, element):

self.array.append(element) def insert\_element(self, index, element):

self.array.insert(index, element) def reverse\_array(self): self.array.rever[se()](https://www.studocu.com/in?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=ii-btech-i-semester-r23-python-programming-lab-manual)

# Example usage: arr = ArrayOperations() arr.create\_array([1, 2, 3, 4, 5]) arr.display\_array() arr.append\_element(6) arr.display\_array() arr.insert\_element(2, 10) arr.display\_array() arr.reverse\_array() arr.display\_array()

**5: Add, transpose, and multiply two matrices**

import numpy as np def matrix\_operations(): # Creating matrices matrix1 = np.array([[1, 2], [3, 4]]) matrix2 = np.array([[5, 6], [7, 8]])

# Addition addition = matrix1 + matrix2 print("Matrix Addition:") print(addition) # Transpose transpose1 = np.transpose(matrix1) transpose2 = np.transpose(matrix2) print("\nMatrix Transpose:") print("Matrix 1:") print(transpose1) print("Matrix 2:") print(transpose2)

# Multiplication

multiplication = np.dot(matrix1, matrix2) print("\nMatrix Multiplication:") print(multiplication) # Example usage: matrix\_operations()

**6: Class representing shapes with area and perimeter methods**

import math class Shape: def area(self):

pass def perimeter(self): pass class Circle(Shape): def \_\_init\_\_(self, radius): self.radius = radius def area(self):

return math.pi \* self.radius \*\* 2 def perimeter(self):

return 2 \* math.pi \* self.radius class Triangle(Shape): def \_\_init\_\_(self, base, height, side1, side2):

self.base = base self.height = height self.side1 = side1 self.side2 = side2 def area(self):

return 0.5 \* sel[f.base \* self.height](https://www.studocu.com/in?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=ii-btech-i-semester-r23-python-programming-lab-manual)

def perimeter(self):

return self.base + self.side1 + self.side2 class Square(Shape): def \_\_init\_\_(self, side): self.side = side def area(self):

return self.side \*\* 2 def perimeter(self): return 4 \* self.side # Example usage: circle = Circle(5) print("Circle Area:", circle.area()) print("Circle Perimeter:", circle.perimeter()) triangle = Triangle(3, 4, 5, 5) print("\nTriangle Area:", triangle.area()) print("Triangle Perimeter:", triangle.perimeter()) square = Square(4) print("\nSquare Area:", square.area()) print("Square Perimeter:", square.perimeter()) **UNIT – 5**

1: Check whether a JSON string contains complex object or not

import json def has\_complex\_object(json\_str): try:

data = json.loads(json\_str) for value in data.values(): if isinstance(value, (list, dict)):

return True return False except json.JSONDecodeError: return False # Example usage: json\_str1 = '{"name": "John", "age": 30, "city": "New York"}'

json\_str2 = '{"name": "Jane", "age": 25, "contacts": {"email": "jane@example.com", "phone": "123-

456-7890"}}'

print(has\_complex\_object(json\_str1)) # False print(has\_complex\_object(json\_str2)) # True

2: Demonstrate NumPy arrays creation using `array()` function

import numpy as np def create\_numpy\_array():

arr = np.array([1, 2, 3, 4, 5]) return arr

# Example usage:

arr = create\_numpy\_array() print("NumPy Array[:", arr)](https://www.studocu.com/in?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=ii-btech-i-semester-r23-python-programming-lab-manual)  3: Demonstrate use of `ndim`, `shape`, `size`, `dtype` in NumPy array

import numpy as np def numpy\_array\_properties():

arr = np.array([[1, 2, 3], [4, 5, 6]]) print("Array:") print(arr)

print("Number of dimensions:", arr.ndim) print("Shape (rows, columns):", arr.shape) print("Size (total number of elements):", arr.size) print("Data type of elements:", arr.dtype) # Example usage:

numpy\_array\_properties()

4: Demonstrate basic slicing, integer and Boolean indexing in NumPy array import numpy as np def numpy\_array\_operations(): arr = np.array([1, 2, 3, 4, 5])

# Basic slicing print("Slicing examples:") print(arr[1:4]) # [2 3 4] print(arr[:3]) # [1 2 3] print(arr[2:]) # [3 4 5] # Integer indexing print("\nInteger indexing:") print(arr[[0, 2, 4]]) # [1 3 5] # Boolean indexing print("\nBoolean indexing:") print(arr[arr > 2]) # [3 4 5] # Example usage:

numpy\_array\_operations()

5: Find min, max, sum, cumulative sum of array using NumPy import numpy as np def numpy\_array\_stats():

arr = np.array([1, 2, 3, 4, 5]) print("Array:", arr)

print("Minimum value:", np.min(arr)) print("Maximum value:", np.max(arr)) print("Sum of array elements:", np.sum(arr)) print("Cumulative sum of array elements:", np.cumsum(arr)) # Example usage: numpy\_array\_stats()

6: Create a dictionary, convert it to a Pandas DataFrame, and explore data import pandas as pd def create\_dataframe\_from\_dict():

data = {

'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],

'Age': [25, 30, 35, 28, 32],

'City': ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Boston'],

'Salary': [50000, 60000, 75000, 45000, 80000]

}

df = pd.DataFrame(data) return df

# Example usage:

df = create\_dataframe\_from\_dict()

# a) Apply head() function print("Head of DataFrame:") print(df.head())

# b) Data selection operations print("\nData selection operations:") print("Selecting column 'Name':") print(df['Name'])

print("\nFiltering based on age > 30:") print(df[df['Age'] > 30])

7: Plotting scatter and line plot using Matplotlib with Pandas DataFrame import matplotlib.pyplot as plt

# Assuming `df` is the DataFrame from 6

# Scatter plot plt.figure(figsize=(8, 6))

plt.scatter(df['Age'], df['Salary'], color='blue', label='Age vs Salary') plt.title('Scatter Plot: Age vs Salary') plt.xlabel('Age') plt.ylabel('Salary') plt.legend() plt.grid(True) plt.show()

# Line plot

plt.figure(figsize=(8, 6))

plt.plot(df['Age'], df['Salary'], marker='o', color='green', linestyle='-', linewidth=2, markersize=8) plt.title('Line Plot: Age vs Salary') plt.xlabel('Age') plt.ylabel('Salary') plt.grid(True) plt.show()

# OUTPUTS

## UNIT-1

1. **Find the largest element among three numbers**

Output

Enter first number: 10

Enter second number: 25

Enter third number: 15

The largest number among 10.0, 25.0, and 15.0 is 25.0.

1. **Display all prime numbers within an interval**

Output

Enter start of interval: 10

Enter end of interval: 50

Prime numbers between 10 and 50 are: [11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]

1. **Swap two numbers without using a temporary variable**

Output

Enter first number: 5

Enter second number: 10

After swapping: a = 10.0, b = 5.0

1. **Demonstrate the following operators in Python with suitable examples**
2. **Arithmetic Operators**

Enter first number for arithmetic operations: 10

Enter second number for arithmetic operations: 3

Addition: 13.0

Subtraction: 7.0

Multiplication: 30.0

Division: 3.3333333333333335

Modulus: 1.0

Exponentiation: 1000.0

Floor Division: 3.0

1. **Relational Operators**

Enter first number for relational operations: 10 Enter second number for relational operations: 3 a == b: False a != b: True a > b: True a < b: False a >= b: True a <= b: False

1. **Assignment Operators**

Enter a number for assignment operations: 10

Enter another number for assignment operations: 3 a += b: 13.0 a -= b: 10.0 a \*= b: 30.0 a /= b: 10.0 a %= b: 1.0 a \*\*= b: 1.0 a //= b: 0.0

1. **Logical Operators**

Enter 1 for True or 0 for False for logical operations (a): 1 Enter 1 for True or 0 for False for logical operations (b): 0 a and b: False a or b: True not a: False

1. **Bitwise Operators**

Enter an integer for bitwise operations (a): 10 Enter another integer for bitwise operations (b): 3 a & b: 2 a | b: 11

a ^ b: 9 ~a: -11 a << 2: 40 a >> 2: 2

1. **Ternary Operator**

Enter first number for ternary operation: 10 Enter second number for ternary operation: 20 b is greater

1. **Membership Operators**

Enter a list of numbers separated by space: 1 2 3 4 5

Enter a number to check membership in the list: 3 Enter another number to check membership in the list: 10

b in a: True c not in a: True

**viii) Identity Operators** a is b: True a is c: False a == c: True

1. **Add and multiply complex numbers**

Output

Enter the real part of first complex number: 2

Enter the imaginary part of first complex number: 3

Enter the real part of second complex number: 4

Enter the imaginary part of second complex number: 5

Addition of (2+3j) and (4+5j) is (6+8j)

Multiplication of (2+3j) and (4+5j) is (-7+22j)

1. **Print multiplication table of a given number**

Output

Enter a number to print its multiplication table: 5

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

5 x 4 = 20

5 x 5 = 25

5 x 6 = 30

5 x 7 = 35

5 x 8 = 40

5 x 9 = 45

5 x 10 = 50

## UNIT-2

1. **Define a function with multiple return values**

**Output**

**Enter first number: 10**

**Enter second number: 2**

**Sum: 12.0, Difference: 8.0, Product: 20.0, Quotient: 5.0**

1. **Define a function using default arguments**

**Output**

**Enter your name: Alice Enter a greeting (optional): Hi Hi, Alice!**

**Enter your name: Bob Enter a greeting (optional):**

**Hello, Bob!**

1. **Find the length of the string without using any library functions**

**Output**

**Enter a string: OpenAI**

**The length of the string 'OpenAI' is 6.**

1. **Check if the substring is present in a given string or not**

**Output**

**Enter the main string: Hello, world!**

**Enter the substring: world**

**The substring 'world' is present in 'Hello, world!'.**

**Enter the main string: Hello, world!**

**Enter the substring: OpenAI**

**The substring 'OpenAI' is not present in 'Hello, world!'.**

1. **Perform operations on a list: addition, insertion, slicing**

**Output**

**Enter a list of numbers separated by space: 1 2 3 4 5**

**Updated List: [1, 20, 2, 3, 4, 5, 10]**

**Sliced List: [20, 2, 3]**

1. **Perform any 5 built-in functions by taking any list**

**Output**

**Enter a list of numbers separated by space: 3 1 4 1 5 9**

**Length: 6**

**Max Value: 9**

**Min Value: 1**

**Sum: 23**

**Sorted List: [1, 1, 3, 4, 5, 9]**

## UNIT-3

1. **Create and concatenate tuples**

**# Output**

**Concatenated Tuple: ('Alice', 21, '123 Maple Street', 'XYZ College', 'Bob', 22, '456 Oak Avenue', 'ABC University')**

1. **Count the number of vowels in a string (No control flow allowed)**

**# Output**

**Enter a string: OpenAI**

**Number of vowels: 3**

1. **Check if a given key exists in a dictionary or not**

**# Output**

**Enter a key to check: age**

**The key 'age' exists in the dictionary.**

**Enter a key to check: college**

**The key 'college' does not exist in the dictionary.**

1. **Add a new key-value pair to an existing dictionary**

**# Output**

**Enter a key to add: college**

**Enter a value to add: XYZ College**

**Updated Dictionary: {'name': 'Alice', 'age': 21, 'address': '123 Maple Street', 'college': 'XYZ College'}**

1. **Sum all the items in a given dictionary**

**# Output**

**Sum of all items: 600**

**UNIT -4**

**1: Sort words in a file and put them in another file**

**Input (`input.txt`):**

**Hello World apple Banana**

**Orange grape**

**Output (`output.txt`):**

**apple banana grape hello orange world**

**2: Print each line of a file in reverse order Input (`input.txt`):**

**First line**

**Second line**

**Third line**

**Output:**

**Third line**

**Second line**

**First line**

**3: Compute number of characters, words, and lines in a file**

**Input (`input.txt`):**

**This is line 1.**

**This is line 2 with more words.**

**And this is line 3.**

**Output:**

**Number of lines: 3**

**Number of words: 16**

**Number of characters: 63**

**4: Operations on an array (create, display, append, insert, reverse)**

**Output:**

**Array: [1, 2, 3, 4, 5]**

**Array: [1, 2, 3, 4, 5, 6]**

**Array: [1, 2, 10, 3, 4, 5, 6] Array: [6, 5, 4, 3, 10, 2, 1]**

**5: Add, transpose, and multiply two matrices**

**Output:**

**Matrix Addition:**

**[[ 6 8]**

**[10 12]]**

**Matrix Transpose:**

**Matrix 1:**

**[[1 3]**

**[2 4]]**

**Matrix 2:**

**[[5 7]**

**[6 8]]**

**Matrix Multiplication:**

**[[19 22]**

**[43 50]]**

**6: Class representing shapes with area and perimeter methods**

**Output:**

**Circle Area: 78.53981633974483**

**Circle Perimeter: 31.41592653589793**

**Triangle Area: 6.0**

**Triangle Perimeter: 13**

**Square Area: 16**

**Square Perimeter: 16**