

S.NO.	DATE	TITLE	PAGE NO.
1	31-01-2025	Develop Python programs using simple Input/Output operations	
1.1		Printing your Name	3
1.2		Printing your Age Next Year	4
2	31-01-2025	Develop Programs using operators and expressions	
2.1		Convert Celsius to Fahrenheit	5
2.2		Find Simple Interest	6
3	07-02-2025	Write Python programs using control statements	
3.1		Leap Year or Not	7
3.2		Day of the Week	8
4	07-02-2025	Develop Python programs to generate series and patterns using control statements	
4.1		Hollow Star Pyramid	9
4.2		Fibonacci Series	10
5	14-02-2025	Develop Python programs using simple functions and recursion	
5.1		Sum of First N Natural Numbers	11
5.2		G.C.D. of Two Numbers	12
6	14-02-2025	Write Python programs for operating on strings and string handling functions	
6.1		Count of different characters in a string	13
6.2		Reversing each word in a sentence	14
7	28-02-2025	Develop Python programs using Lists, Nested Lists, and List Comprehensions	
7.1		Rotating a list	15
7.2		Filtering a list	16
8	14-03-2025	Develop Python Programs using Tuples, Nested Tuples, and Tuple Comprehension	
8.1		N number of minimum and maximum values	17
8.2		Flattening a nested tuple	18

9	21-03-2025	Write Python programs creating sets and performing set operations	
9.1		Common Students	19
9.2		Unique Elements	20
10	04-04-2025	Develop Python programs using Dictionary, Nested Dictionary, and Dictionary Comprehension	
10.1		Even Squares	21
10.2		Top Student by Average	22
11	11-04-2025	Design Python programs to handle errors and exceptions	
11.1		Valid input	23
11.2		Division of two numbers	24
12	11-04-2025	Write Python programs with multiple handlers for exceptions	
12.1		Division of two numbers	25
12.2		Square root of number	26
13	25-04-2025	Write Python programs to read, create, and update text files	
13.1		Copy one File into Another	27
13.2		Count of lines and words in a file	29

Exp. No.: 1

Date: 31.01.2025

DEVELOP PYTHON PROGRAMS USING SIMPLE INPUT/OUTPUT OPERATIONS

1.1 – Printing your Name

Aim:

Write a Python Program to print your name

Source Code:

```
1 name = input("Enter Your Name: ")
2 print("Your name is", name)
```

Output:

```
Enter Your Name: Shiva Prakash
Your name is Shiva Prakash
```

Result:

The Python Program to print your name has been successfully written and executed.

1.2 – Printing your Age Next Year

Aim:

Write a Python Program to print your age next year

Source Code:

```
1 age = int(input("Enter your age: "))  
2 print("You will be", age + 1, "next year!")
```

Output:

```
Enter your age: 18  
You will be 19 next year!
```

Result:

The Python Program to print your age next year has been successfully written and executed.

Exp. No.: 2

Date: 31.01.2025

DEVELOP PROGRAMS USING OPERATORS AND EXPRESSIONS

2.1 – Convert Celsius to Fahrenheit

Aim:

Write a Python Program to convert Celsius to Fahrenheit.

Source Code:

```
1 celsius = float(input("Enter Temperature in Celsius: "))
2 fahrenheit = (celsius * 1.8) + 32
3 print("Temperature in Fahrenheit:", fahrenheit)
```

Output:

```
Enter Temperature in Celsius: 32
Temperature in Fahrenheit: 89.6
```

Result:

The Python Program to convert Celsius to Fahrenheit has been successfully written and executed.

2.2 – Find Simple Interest

Aim:

Write a Python Program to Find Simple Interest.

Source Code:

```
1 principal = float(input("Enter Principal Amount (Rs.): "))
2 rate = float(input("Enter Rate of Interest (%): "))
3 time = float(input("Enter Time in Years: "))
4
5 interest = (principal * rate * time)/100
6 print("Amount after", time, "years:", principal + interest)
```

Output:

```
Enter Principal Amount (Rs.): 10000
Enter Rate of Interest (%): 5
Enter Time in Years: 10
Amount after 10.0 years: 15000.0
```

Result:

The Python Program to Find Simple Interest has been successfully written and executed.

Exp. No.: 3

Date: 07.02.2025

WRITE PYTHON PROGRAMS USING CONTROL STATEMENTS

3.1 – Leap Year or Not

Aim:

Write a Python Program to Check Whether the Year is Leap Year or Not.

Source Code:

```
1 year = int(input("Enter Year: "))
2
3 if (year % 400 == 0):
4     print(year, "is a leap year")
5 elif (year % 4 == 0 and year % 100 != 0):
6     print(year, "is a leap year")
7 else:
8     print(year, "is not a leap year")
```

Output:

Enter Year: 2000	Enter Year: 1000
2000 is a leap year	1000 is not a leap year

Result:

The Python Program to Check Whether the Year is Leap Year or Not has been successfully written and executed.

3.2 – Day of the Week

Aim:

Write a Python Program to Find the Day of the Week.

Source Code:

```
1 day = int(input("Enter Day (1-7): "))
2 if day == 1:
3     print("Monday")
4 elif day == 2:
5     print("Tuesday")
6 elif day == 3:
7     print("Wednesday")
8 elif day == 4:
9     print("Thursday")
10 elif day == 5:
11     print("Friday")
12 elif day == 6:
13     print("Saturday")
14 elif day == 7:
15     print("Sunday")
16 else:
17     print("Between 1 to 7")
```

Output:

```
Enter Day (1-7): 5
Friday
```

Result:

The Python Program to Find the Day of the Week has been successfully written and executed.

Exp. No.: 4

Date: 07.02.2025

PROGRAMS USING FOR, WHILE, DO-WHILE LOOPS AND NESTED LOOPS

4.1 – Print A Hollow Star Pyramid

Aim:

Write a Python Program to Print a Hollow Star Pyramid.

Source Code:

```
1 num = int(input("Enter number of rows: ")) + 1
2 for i in range(num):
3     for j in range(num - i):
4         print(" ", end="")
5     for k in range(2 * i - 1):
6         if (k == 0 or k == 2 * i - 2) or i == num - 1:
7             print("* ", end="")
8         else:
9             print("  ", end="")
10    print()
```

Output:

```
Enter number of rows: 5

      *
     * *
    *   *
   *     *
  *       *
 *         *
* * * * *
* * * * *
```

Result:

The Python Program to Print a Hollow Star Pyramid has been successfully written and executed.

4.2 – Fibonacci Series

Aim:

Write a Python Program to print the Fibonacci Series.

Source Code:

```
1 num = int(input("Enter number of terms: "))
2
3 a, b = 0, 1
4 for _ in range(num):
5     print(a, end=" ")
6     a, b = b, a + b
```

Output:

```
Enter number of terms: 10
0 1 1 2 3 5 8 13 21 34
```

Result:

The Python Program to print the Fibonacci Series has been successfully written and executed.

Exp. No.: 5

Date: 14.02.2025

DEVELOP PYTHON PROGRAMS USING SIMPLE FUNCTIONS AND RECURSION

5.1 – Sum of first N Natural Numbers

Aim:

Write a Python Program to find the sum of first N Natural Numbers.

Source Code:

```
1 def sum(num):  
2     if num <= 1:  
3         return num  
4     return num + sum(num - 1)  
5  
6  
7 nat_nums = int(input("Enter number: "))  
8 print("Sum of first", nat_nums, "natural numbers is", sum(nat_nums))
```

Output:

```
Enter number: 12  
Sum of first 12 natural numbers is 78
```

Result:

The Python Program to sum of first N Natural Numbers has been successfully written and executed.

5.2 – G.C.D. of Two Numbers

Aim:

Write a Python Program to find G.C.D. of Two Numbers.

Source Code:

```
1 def gcd(a, b):
2     if a > b:
3         x, y = a, b
4     else:
5         x, y = b, a
6
7     if y == 0:
8         return x
9     else:
10        return gcd(x % y, y)
11
12
13 num1 = int(input("Enter Number1: "))
14 num2 = int(input("Enter Number2: "))
15 print("The GCD of", num1, "and", num2, "is", gcd(num1, num2))
```

Output:

```
Enter Number1: 72
Enter Number2: 160
The GCD of 72 and 160 is 8
```

Result:

The Python Program to find G.C.D. of Two Numbers has been successfully written and executed.

Exp. No.: 6

Date: 14.02.2025

WRITE PYTHON PROGRAMS FOR OPERATING ON STRINGS AND STRING HANDLING FUNCTIONS

6.1 – Count of different characters in a string

Aim:

Write a Python Program to count different characters in a string.

Source Code:

```
1 text = input("Enter a string: ")
2
3 vowels = consonants = digits = spaces = 0
4
5 for char in text:
6     if char.lower() in "aeiou":
7         vowels += 1
8     elif char.isalpha():
9         consonants += 1
10    elif char.isdigit():
11        digits += 1
12    elif char.isspace():
13        spaces += 1
14
15 print("Vowels:", vowels)
16 print("Consonants:", consonants)
17 print("Digits:", digits)
18 print("Spaces:", spaces)
```

Output:

```
Enter a string: DØMS makes better stationary items
Vowels: 10
Consonants: 19
Digits: 1
Spaces: 4
```

Result:

The Python Program to count different characters in a string has been successfully written and executed.

6.2 – Reversing each word in a sentence

Aim:

Write a Python Program to reverse each word in a sentence.

Source Code:

```
1 sentence = input("Enter a sentence: ")
2
3 words = sentence.split()
4 reversed_words = [word[::-1] for word in words]
5
6 new_sentence = " ".join(reversed_words)
7 print("Reversed words sentence:", new_sentence)
```

Output:

```
Enter a sentence: DØMS makes better stationary items
Reversed words sentence: SMØD sekam retteb yranoitats smeti
```

Result:

The Python Program to reverse each word in a sentence has been successfully written and executed.

Exp. No.: 7

Date: 28.02.2025

DEVELOP PYTHON PROGRAMS USING LISTS, NESTED LISTS, AND LIST COMPREHENSIONS

7.1 – Rotating a list

Aim:

Write a Python Program to rotate a list.

Source Code:

```
1 lst = input("Enter elements: ").split()
2 rot_num = int(input("Enter rotation number: "))
3 split_num = rot_num % len(lst)
4 rotated_lst = lst[split_num:] + lst[:split_num]
5 print("Rotated List:", rotated_lst)
```

Output:

```
Enter elements: 43 83 22 84 12 03 1
Enter rotation number: 3
Rotated List: ['84', '12', '03', '1', '43', '83', '22']
```

Result:

The Python Program to rotate a list has been successfully written and executed.

7.2 – Filtering a list

Aim:

Write a Python Program to filter the words containing the letter 'a' from the list.

Source Code:

```
1 lst = input("Enter elements: ").split()
2 filtered_lst = [ele for ele in lst if "a" not in ele]
3 print("Filtered List:", filtered_lst)
```

Output:

```
Enter elements: Hola World This is a toast!
Filtered List: ['World', 'This', 'is']
```

Result:

The Python Program to filter the words containing the letter 'a' from the list has been successfully written and executed.

Exp. No.: 8

Date: 14.03.2025

DEVELOP PYTHON PROGRAMS USING TUPLES, NESTED TUPLES, AND TUPLE COMPREHENSION

8.1 – N number of minimum and maximum values

Aim:

Write a Python Program to find N number of minimum and maximum values.

Source Code:

```
1 tpl = tuple(input("Enter elements: ").split())
2 tpl = tuple(map(int, tpl))
3 num = int(input("Enter number of results: "))
4
5 sorted_lst = sorted(tpl)
6 values = []
7 for i in range(num):
8     values.append(sorted_lst[i])
9 for i in range((len(sorted_lst) - num), len(sorted_lst)):
10    values.append(sorted_lst[i])
11
12 print("Min Values:", values[:num])
13 print("Max values:", values[num:])
```

Output:

```
Enter elements: 31 4 09 33 2 -42 43
Enter number of results: 2
Min Values: [-42, 2]
Max values: [33, 43]
```

Result:

The Python Program to find the N number of minimum and maximum values has been successfully written and executed.

8.2 – Flattening a nested tuple

Aim:

Write a Python Program to flatten a nested tuple.

Source Code:

```
1 def flatten(tpl):
2     flat_lst = []
3     for element in tpl:
4         if not isinstance(element, tuple):
5             flat_lst.append(element)
6         else:
7             flat_lst += flatten(element)
8     return tuple(flat_lst)
9
10
11 nestedTuple = (5, (2, 5), ((1, 5), (4, 7)), (2, (4, 3)))
12
13 flattenTuple = flatten(nestedTuple)
14 print("The flattened tuple : " + str(flattenTuple))
```

Output:

```
The flattened tuple : (5, 2, 5, 1, 5, 4, 7, 2, 4, 3)
```

Result:

The Python Program to flatten a nested tuple has been successfully written and executed.

Exp. No.: 9

Date: 21.03.2025

WRITE PYTHON PROGRAMS CREATING SETS AND PERFORMING SET OPERATIONS

9.1 – Common Students

Aim:

Write a Python Program to find the common students in class 1 and 2.

Source Code:

```
1 set1 = set(input("Enter students in class 1: ").split())
2 set2 = set(input("Enter students in class 2: ").split())
3
4 commonSet = set1.intersection(set2)
5 print("Students in both classes:", commonSet)
```

Output:

```
Enter students in class 1: Rohit Shiva Somu Chandran
Enter students in class 2: Shiva Dharshan Chandran Dinesh
Students in both classes: {'Shiva', 'Chandran'}
```

Result:

The Python Program to find the common students in class 1 and 2 has been successfully written and executed.

9.2 – Unique Elements

Aim:

Write a Python Program to find the unique elements in a list.

Source Code:

```
1 lst = input("Enter Elements: ").split()
2 unique_lst = sorted(set(lst))
3 print("Unique Items in List:", unique_lst)
```

Output:

```
Enter Elements: 73 82 93 73 91 91 43 12 73
Unique Items in List: ['12', '43', '73', '82', '91', '93']
```

Result:

The Python Program to find the unique elements in a list has been successfully written and executed.

Exp. No.: 10

Date: 04.04.2025

DEVELOP PYTHON PROGRAMS USING DICTIONARY, NESTED DICTIONARY, AND DICTIONARY COMPREHENSION

10.1 – Even Squares

Aim:

Write a Python Program to store the even squares in a dictionary.

Source Code:

```
1 num = int(input("Enter Upper Limit: "))
2
3 squares = {x: x**2 for x in range(1, num + 1) if x % 2 == 0}
4 print("Even number squares:", squares)
```

Output:

```
Enter Upper Limit: 10
Even number squares: {2: 4, 4: 16, 6: 36, 8: 64, 10: 100}
```

Result:

The Python Program to store the even squares in a dictionary has been successfully written and executed.

10.2 – Top Student by Average

Aim:

Write a Python Program to find the top student by average.

Source Code:

```
1 students = {  
2     "Alice": {"Math": 85, "Science": 90, "English": 88},  
3     "Bob": {"Math": 78, "Science": 74, "English": 80},  
4     "Charlie": {"Math": 92, "Science": 89, "English": 95},  
5 }  
6  
7 for name, subjects in students.items():  
8     avg = sum(subjects.values()) / len(subjects)  
9     students[name]["Average"] = avg  
10  
11 top_student = max(students, key=lambda x: students[x]["Average"])  
12 print("Top Scorer is", top_student, "with average", students[top_student]["Average"])
```

Output:

```
Top Scorer is Charlie with average 92.0
```

Result:

The Python Program to find the top student by average has been successfully written and executed.

Exp. No.: 11

Date: 11.04.2025

DESIGN PYTHON PROGRAMS TO HANDLE ERRORS AND EXCEPTIONS

11.1 – Valid Input

Aim:

Write a Python Program to check if given input is valid or not.

Source Code:

```
1 try:
2     n = int(input("Enter a number: "))
3 except ValueError:
4     print("Invalid input")
5 else:
6     print("Valid input")
```

Output:

```
Enter a number: 54
Valid input
```

```
Enter a number: No
Invalid input
```

Result:

The Python Program to check if given input is valid or not has been successfully written and executed.

11.2 – Division of two numbers

Aim:

Write a Python Program to divide two numbers with error handling.

Source Code:

```
1 try:
2     dividend = int(input("Enter dividend: "))
3     divisor = int(input("Enter divisor: "))
4     print(dividend / divisor)
5 except ZeroDivisionError:
6     print("Cannot divide by zero")
```

Output:

Enter dividend: 54	Enter dividend: 54
Enter divisor: 6	Enter divisor: 0
9.0	Cannot divide by zero

Result:

The Python Program to divide two numbers with error handling has been successfully written and executed.

Exp. No.: 12

Date: 11.04.2025

DESIGN PYTHON PROGRAMS WITH MULTIPLE HANDLERS FOR EXCEPTIONS

12.1 – Division of two numbers

Aim:

Write a Python Program to divide two numbers with multiple exception handlers.

Source Code:

```
1 try:
2     dividend = int(input("Enter dividend: "))
3     divisor = int(input("Enter divisor: "))
4     print(dividend / divisor)
5 except ZeroDivisionError:
6     print("Cannot divide by zero")
7 except ValueError:
8     print("Invalid input")
9 except Exception as err:
10    print("Unexpected Error:", err)
```

Output:

Enter dividend: 89	Enter dividend: 89	Enter dividend: 89
Enter divisor: 4	Enter divisor: 0	Enter divisor: Zero
22.25	Cannot divide by zero	Invalid input

Result:

The Python Program to divide two numbers with multiple exception handlers has been successfully written and executed.

12.2 – Square root of number

Aim:

Write a Python Program to find the square root of a number with multiple exception handlers.

Source Code:

```
1 def root(x):
2     if x < 0:
3         raise ValueError("No Negative Numbers")
4     else:
5         return x**0.5
6
7
8 try:
9     num = int(input("Enter number: "))
10    print(root(num))
11 except ValueError as err:
12    print("Invalid Input:", err)
13 except Exception as err:
14    print("Unexpected Error:", err)
```

Output:

Enter number: 9 3.0	Enter number: -9 Invalid Input: No Negative Numbers
Enter number: Nine Invalid Input: invalid literal for int() with base 10: 'Nine'	

Result:

The Python Program to find the square root of a number with multiple exception handlers has been successfully written and executed.

Exp. No.: 13

Date: 25.04.2025

WRITE PYTHON PROGRAMS TO READ, CREATE, AND UPDATE TEXT FILES

13.1 – Copy one File into Another

Aim:

Write a Python Program to Copy one File into Another.

Source Code:

```
1 try:
2     filename = input("Enter source filename: ")
3
4     with open(filename, "r") as f:
5         data = f.read()
6
7     filename = input("Enter destination filename: ")
8
9     with open(filename, "w") as f:
10        f.write(data)
11 except FileNotFoundError:
12     print("Error: File not found.")
13 except Exception as e:
14     print("Unexpected error:", e)
15 else:
16     print("Successfully Copied")
```

Input:

12.2.txt:

```
1 Enter number: 9
2 3.0
3
4 Enter number: -9
5 Invalid Input: No Negative Numbers
6
7 Enter number: Nine
8 Invalid Input: invalid literal for int() with base 10: 'Nine'
```

Output:

```
Enter source filename: 12.2.txt  
Enter destination filename: dest.txt  
Successfully Copied
```

dest.txt:

```
1 Enter number: 9  
2 3.0  
3  
4 Enter number: -9  
5 Invalid Input: No Negative Numbers  
6  
7 Enter number: Nine  
8 Invalid Input: invalid literal for int() with base 10: 'Nine'
```

Result:

The Python Program to Copy one File into Another has been successfully written and executed.

13.2 – Count of Lines and Words in a File

Aim:

Write a Python Program to count the number of Lines and Words in a File.

Source Code:

```
1 try:
2     filename = input("Enter filename: ")
3
4     with open(filename, "r") as f:
5         lines = f.readlines()
6
7     line_count = len(lines)
8     word_count = sum(len(line.split()) for line in lines)
9
10    print("Total Lines:", line_count)
11    print("Total Words:", word_count)
12
13 except FileNotFoundError:
14     print("Error: File not found.")
15 except Exception as e:
16     print("Unexpected error:", e)
```

Input:

12.1.txt:

```
1 Enter dividend: 89
2 Enter divisor: 4
3 22.25
4
5 Enter dividend: 89
6 Enter divisor: 0
7 Cannot divide by zero
8
9 Enter dividend: 89
10 Enter divisor: Zero
11 Invalid input
```

Output:

```
Enter filename: 12.1.txt  
Total Lines: 11  
Total Words: 25
```

Result:

The Python Program to count the number of Lines and Words in a File has been successfully written and executed.