Exp. No.: 1 **Date: 31.01.2025**

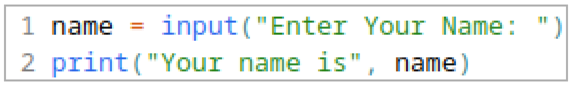
DEVELOP PYTHON PROGRAMS USING SIMPLE INPUT/OUTPUT OPERATIONS

1.1 – Printing your Name

Aim:

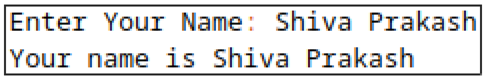
Write a Python Program to print your name

Source Code:



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

Output:



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

Result:

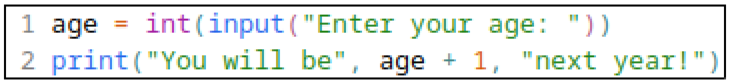
The Python Program to print your name has been successfully written and executed.

1.2 – Printing your Age Next Year

Aim:

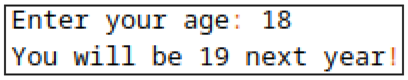
Write a Python Program to print your age next year

Source Code:



```python
age = int(input("Enter your age: "))
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**

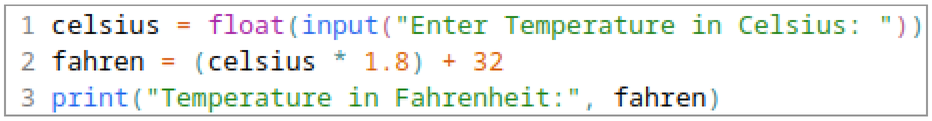
DEVELOP PROGRAMS USING OPERATORS AND EXPRESSIONS

2.1 – Convert Celsius to Fahrenheit

Aim:

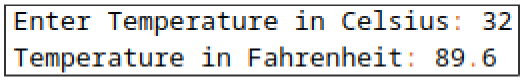
Write a Python Program to convert Celsius to Fahrenheit.

Source Code:



```python
celsius = float(input("Enter Temperature in Celsius: "))
fahren = (celsius * 1.8) + 32
print("Temperature in Fahrenheit:", fahren)
```

Output:



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

Result:

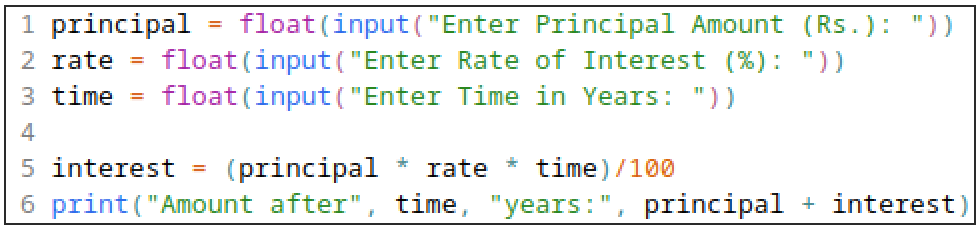
The Python Program to convert Celsius to Fahrenheit has been successfully written and executed.

2.2 – Find Simple Interest

Aim:

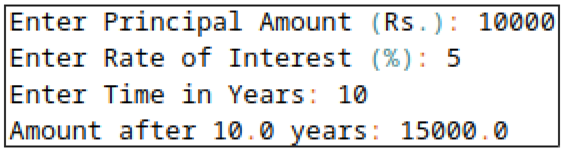
Write a Python Program to Find Simple Interest.

Source Code:



```python
principal = float(input("Enter Principal Amount (Rs.): "))
rate = float(input("Enter Rate of Interest (%): "))
time = float(input("Enter Time in Years: "))

interest = (principal * rate * time)/100
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**

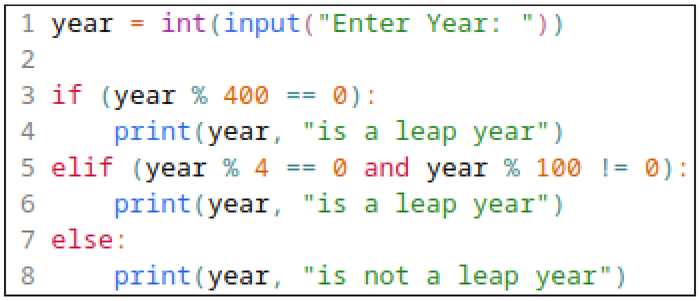
WRITE PYTHON PROGRAMS USING CONTROL STATEMENTS

3.1 – Leap Year or Not

Aim:

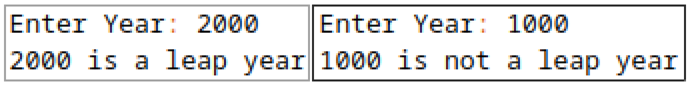
Write a Python Program to Check Whether the Year is Leap Year or Not.

Source Code:



```python
year = int(input("Enter Year: "))

if (year % 400 == 0):
    print(year, "is a leap year")
elif (year % 4 == 0 and year % 100 != 0):
    print(year, "is a leap year")
else:
    print(year, "is not a leap year")
```

Output:



```
Enter Year: 2000
2000 is a leap year
```

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

Result:

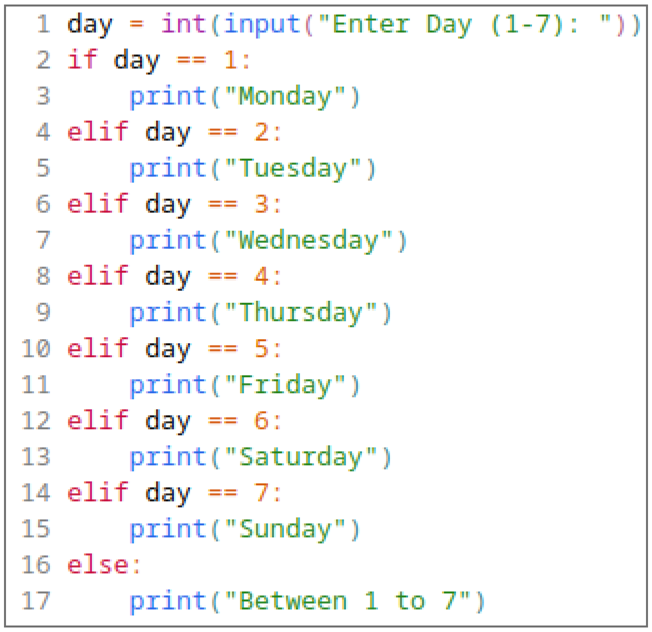
The Python Program to Check Whether the Year is Leap Year or Not has been successfully written and executed.

3.2 – Day of the Week

Aim:

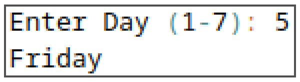
Write a Python Program to Find the Day of the Week.

Source Code:



```python
day = int(input("Enter Day (1-7): "))
if day == 1:
    print("Monday")
elif day == 2:
    print("Tuesday")
elif day == 3:
    print("Wednesday")
elif day == 4:
    print("Thursday")
elif day == 5:
    print("Friday")
elif day == 6:
    print("Saturday")
elif day == 7:
    print("Sunday")
else:
    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**

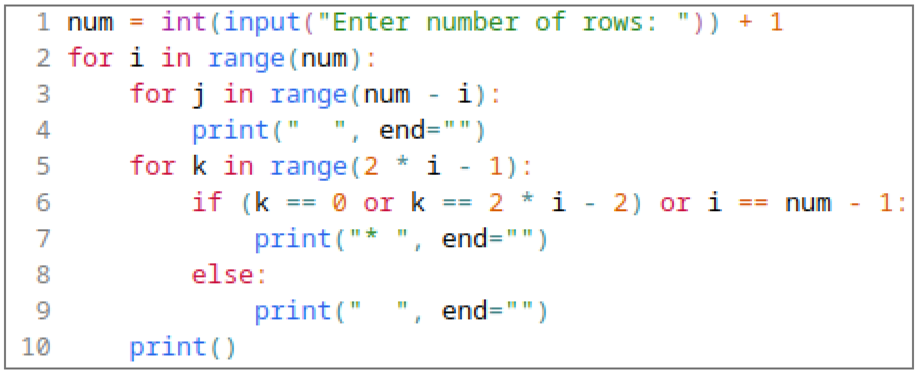
PROGRAMS USING FOR, WHILE, DO-WHILE LOOPS AND NESTED LOOPS

4.1 – Print A Hollow Star Pyramid

Aim:

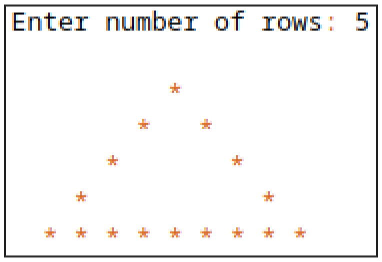
Write a Python Program to Print a Hollow Star Pyramid.

Source Code:



```python
num = int(input("Enter number of rows: ")) + 1
for i in range(num):
    for j in range(num - i):
        print("  ", end="")
    for k in range(2 * i - 1):
        if (k == 0 or k == 2 * i - 2) or i == num - 1:
            print("* ", end="")
        else:
            print("  ", end="")
    print()
```

Output:



```
Enter number of rows: 5

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

Result:

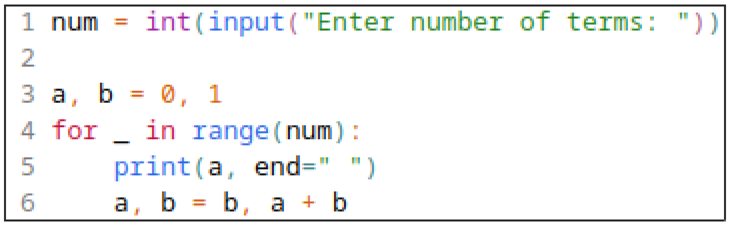
The Python Program to Print a Hollow Star Pyramid has been successfully written and executed.

4.2 – Fibonacci Series

Aim:

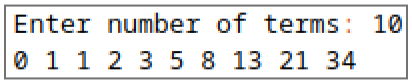
Write a Python Program to print the Fibonacci Series.

Source Code:



```python
num = int(input("Enter number of terms: "))

a, b = 0, 1
for _ in range(num):
    print(a, end=" ")
    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**

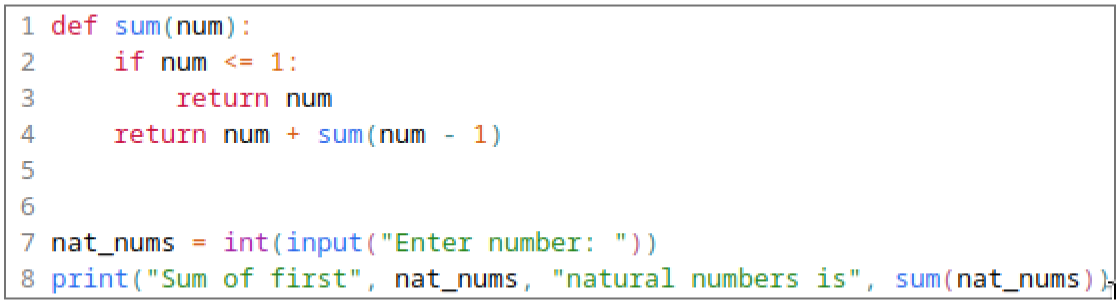
DEVELOP PYTHON PROGRAMS USING SIMPLE FUNCTIONS AND RECURSION

5.1 – Sum of first N Natural Numbers

Aim:

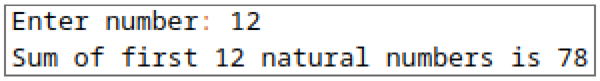
Write a Python Program to find the sum of first N Natural Numbers.

Source Code:



```python
def sum(num):
    if num <= 1:
        return num
    return num + sum(num - 1)


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

```python
def gcd(a, b):
    if a > b:
        x, y = a, b
    else:
        x, y = b, a

    if y == 0:
        return x
    else:
        return gcd(x % y, y)


num1 = int(input("Enter Number1: "))
num2 = int(input("Enter Number2: "))
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**

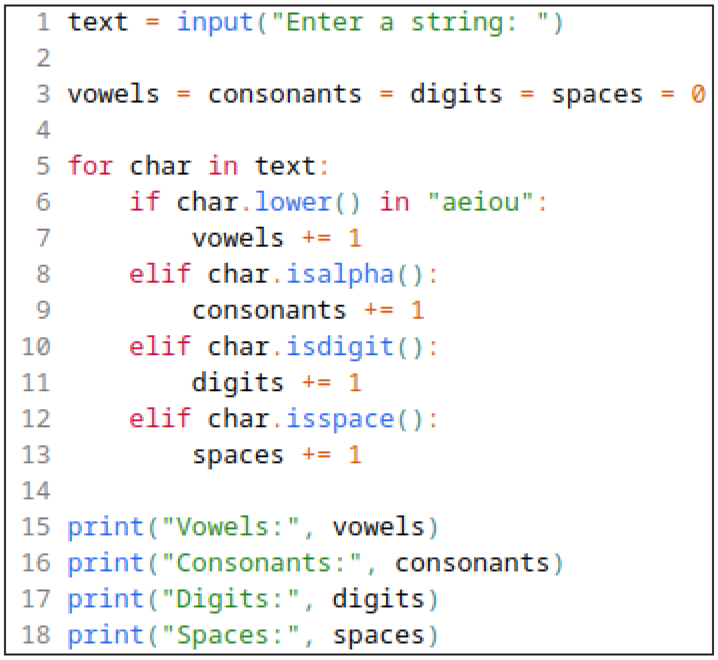
WRITE PYTHON PROGRAMS FOR OPERATING ON STRINGS AND STRING HANDLING FUNCTIONS

6.1 – Count of different characters in a string

Aim:

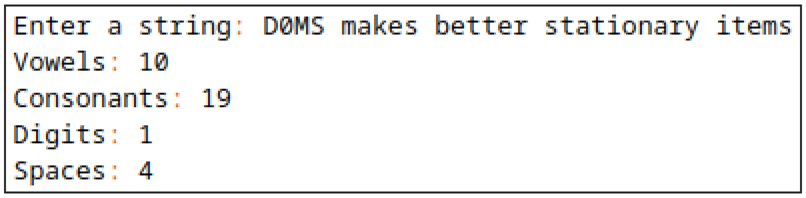
Write a Python Program to count different characters in a string.

Source Code:



```python
text = input("Enter a string: ")

vowels = consonants = digits = spaces = 0

for char in text:
    if char.lower() in "aeiou":
        vowels += 1
    elif char.isalpha():
        consonants += 1
    elif char.isdigit():
        digits += 1
    elif char.isspace():
        spaces += 1

print("Vowels:", vowels)
print("Consonants:", consonants)
print("Digits:", digits)
print("Spaces:", spaces)
```

Output:



```
Enter a string: D0MS makes better stationary items
Vowels: 10
Consonants: 19
Digits: 1
Spaces: 4
```

Result:

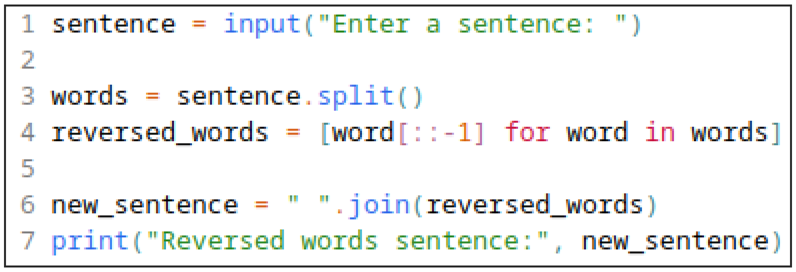
The Python Program to count different characters in a string has been successfully written and executed.

6.2 – Reversing each word in a sentence

Aim:

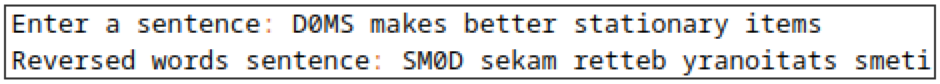
Write a Python Program to reverse each word in a sentence.

Source Code:



```python
sentence = input("Enter a sentence: ")

words = sentence.split()
reversed_words = [word[::-1] for word in words]

new_sentence = " ".join(reversed_words)
print("Reversed words sentence:", new_sentence)
```

Output:



```
Enter a sentence: D0MS makes better stationary items
Reversed words sentence: SM0D 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**

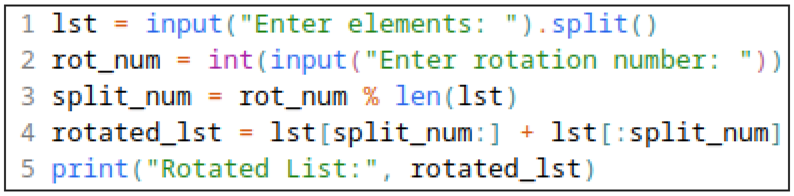
DEVELOP PYTHON PROGRAMS USING LISTS, NESTED LISTS, AND LIST COMPREHENSIONS

7.1 – Rotating a list

Aim:

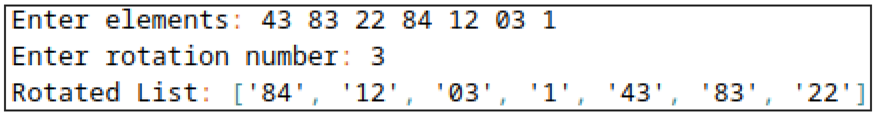
Write a Python Program to rotate a list.

Source Code:



```python
lst = input("Enter elements: ").split()
rot_num = int(input("Enter rotation number: "))
split_num = rot_num % len(lst)
rotated_lst = lst[split_num:] + lst[:split_num]
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:

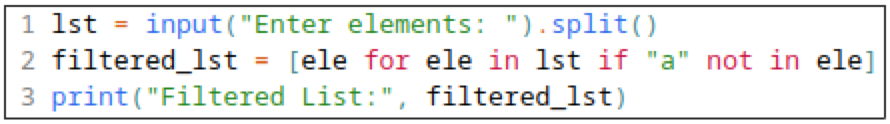
The Python Program to rotate a list has been successfully written and executed.

7.2 – Filtering a list

Aim:

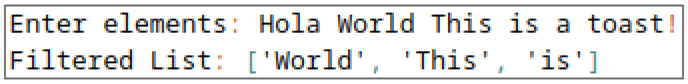
Write a Python Program to filter the words containing the letter ‘a’ from the list.

Source Code:



```python
lst = input("Enter elements: ").split()
filtered_lst = [ele for ele in lst if "a" not in ele]
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**

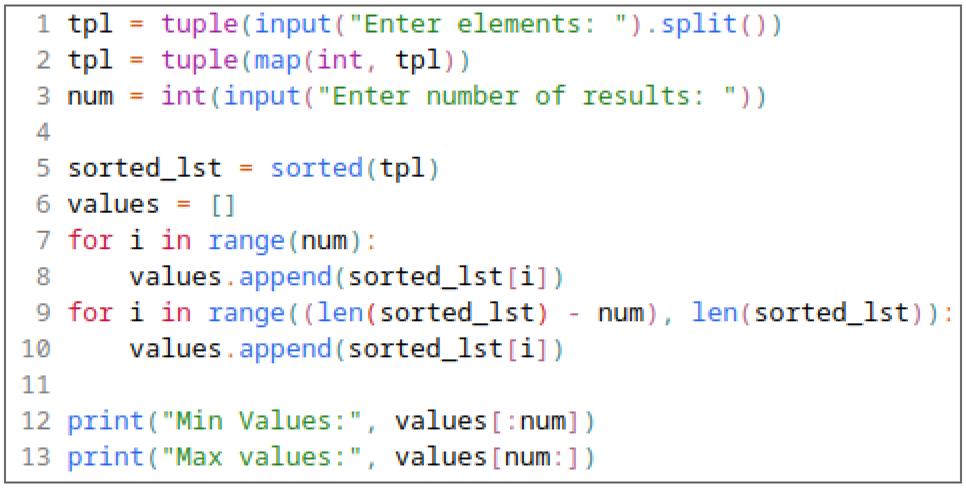
DEVELOP PYTHON PROGRAMS USING TUPLES, NESTED TUPLES, AND TUPLE COMPREHENSION

8.1 – N number of minimum and maximum values

Aim:

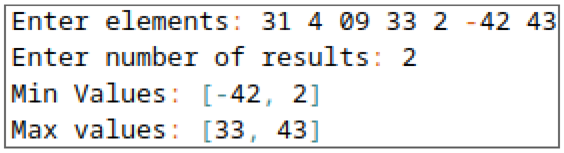
Write a Python Program to find N number of minimum and maximum values.

Source Code:



```python
tpl = tuple(input("Enter elements: ").split())
tpl = tuple(map(int, tpl))
num = int(input("Enter number of results: "))

sorted_lst = sorted(tpl)
values = []
for i in range(num):
    values.append(sorted_lst[i])
for i in range((len(sorted_lst) - num), len(sorted_lst)):
    values.append(sorted_lst[i])

print("Min Values:", values[:num])
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:

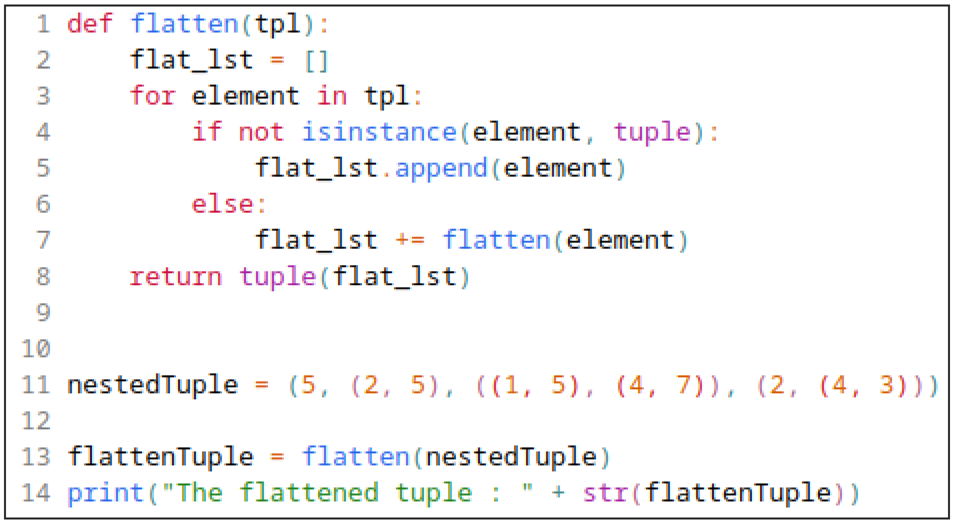
The Python Program to find the N number of minimum and maximum values has been successfully written and executed.

8.2 – Flattening a nested tuple

Aim:

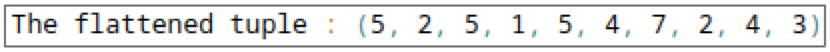
Write a Python Program to flatten a nested tuple.

Source Code:



```python
def flatten(tpl):
    flat_lst = []
    for element in tpl:
        if not isinstance(element, tuple):
            flat_lst.append(element)
        else:
            flat_lst += flatten(element)
    return tuple(flat_lst)


nestedTuple = (5, (2, 5), ((1, 5), (4, 7)), (2, (4, 3)))

flattenTuple = flatten(nestedTuple)
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**

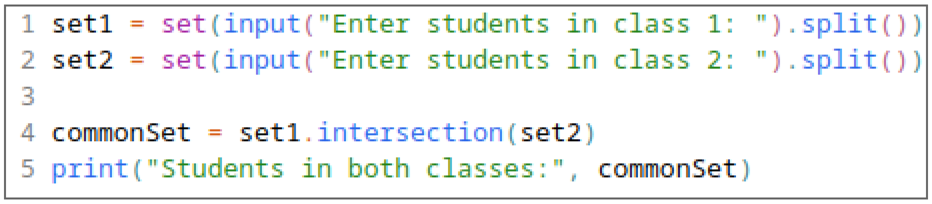
WRITE PYTHON PROGRAMS CREATING SETS AND PERFORMING SET OPERATIONS

10.1 – Common Students

Aim:

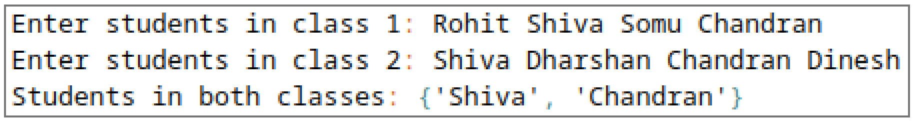
Write a Python Program to find the common students in class 1 and 2.

Source Code:



```python
set1 = set(input("Enter students in class 1: ").split())
set2 = set(input("Enter students in class 2: ").split())

commonSet = set1.intersection(set2)
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:

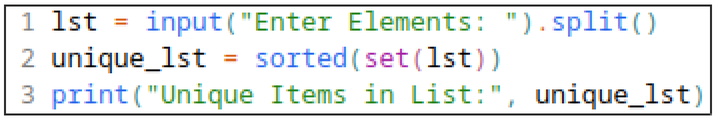
The Python Program to find the common students in class 1 and 2 has been successfully written and executed.

9.2 – Unique Elements

Aim:

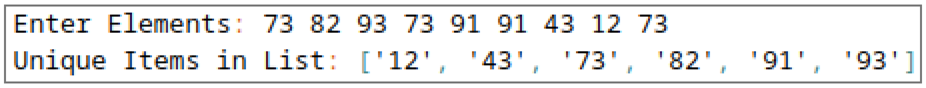
Write a Python Program to find the unique elements in a list.

Source Code:



```python
lst = input("Enter Elements: ").split()
unique_lst = sorted(set(lst))
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**

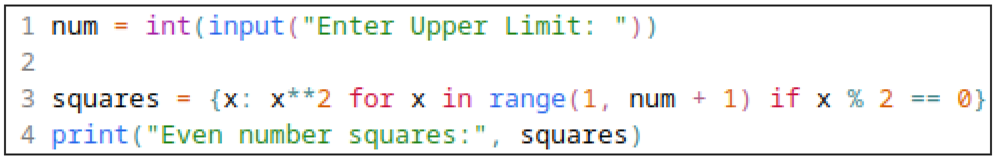
DEVELOP PYTHON PROGRAMS USING DICTIONARY, NESTED DICTIONARY, AND DICTIONARY COMPREHENSION

10.1 – Even Squares

Aim:

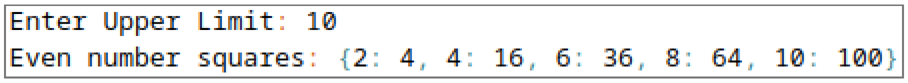
Write a Python Program to store the even squares in a dictionary.

Source Code:



```python
num = int(input("Enter Upper Limit: "))

squares = {x: x**2 for x in range(1, num + 1) if x % 2 == 0}
print("Even number squares:", squares)
```

Output:



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

Result:

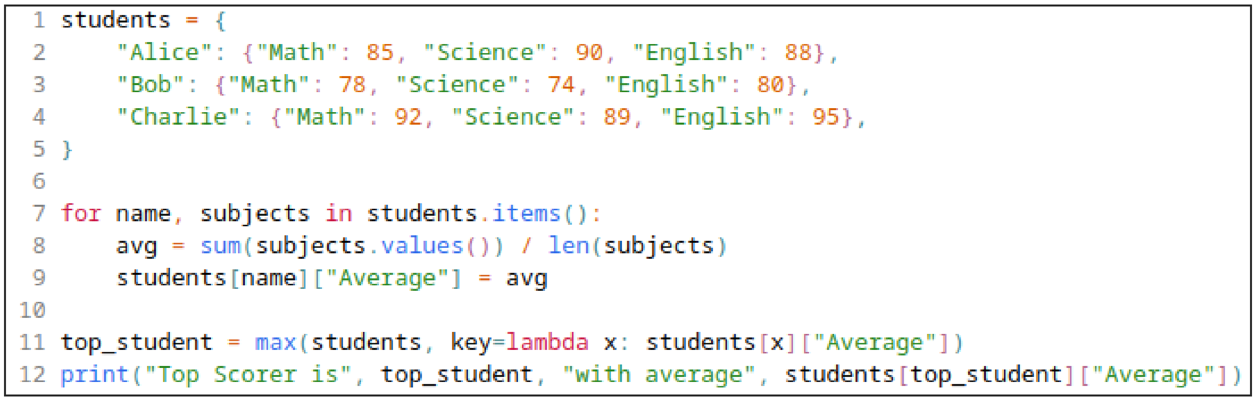
The Python Program to store the even squares in a dictionary has been successfully written and executed.

11.2 – Top Student by Average

Aim:

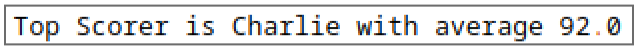
Write a Python Program to find the top student by average.

Source Code:



```python
students = {
    "Alice": {"Math": 85, "Science": 90, "English": 88},
    "Bob": {"Math": 78, "Science": 74, "English": 80},
    "Charlie": {"Math": 92, "Science": 89, "English": 95},
}

for name, subjects in students.items():
    avg = sum(subjects.values()) / len(subjects)
    students[name]["Average"] = avg

top_student = max(students, key=lambda x: students[x]["Average"])
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**

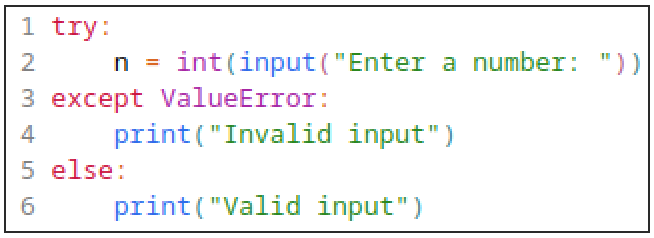
DESIGN PYTHON PROGRAMS TO HANDLE ERRORS AND EXCEPTIONS

11.1 – Valid Input

Aim:

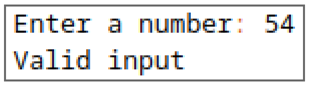
Write a Python Program to check if given input is valid or not.

Source Code:

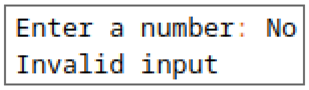


```python
try:
    n = int(input("Enter a number: "))
except ValueError:
    print("Invalid input")
else:
    print("Valid input")
```

Output:



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



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

Result:

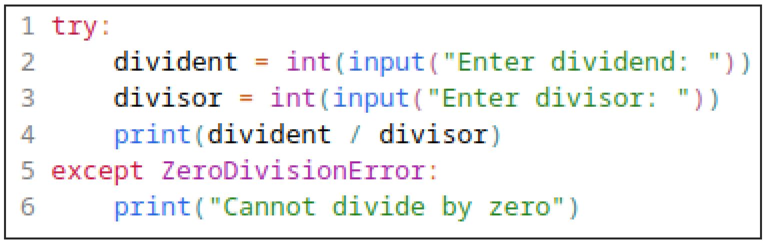
The Python Program to check if given input is valid or not has been successfully written and executed.

11.2 – Division of two numbers

Aim:

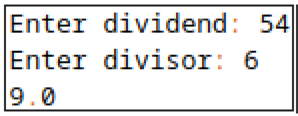
Write a Python Program to divide two numbers with error handling.

Source Code:

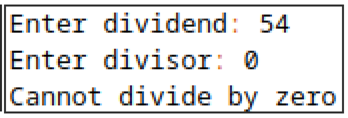


```python
try:
    divident = int(input("Enter dividend: "))
    divisor = int(input("Enter divisor: "))
    print(divident / divisor)
except ZeroDivisionError:
    print("Cannot divide by zero")
```

Output:



```
Enter dividend: 54
Enter divisor: 6
9.0
```



```
Enter dividend: 54
Enter divisor: 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**

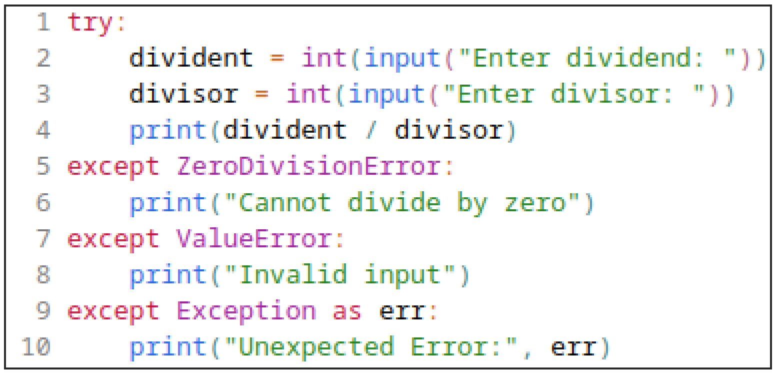
DESIGN PYTHON PROGRAMS WITH MULTIPLE HANDLERS FOR EXCEPTIONS

12.1 – Division of two numbers

Aim:

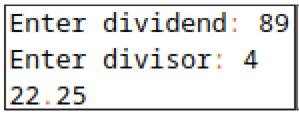
Write a Python Program to divide two numbers with multiple exception handlers.

Source Code:

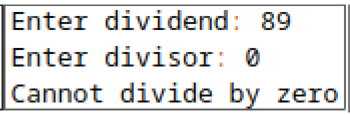


```python
try:
    divident = int(input("Enter dividend: "))
    divisor = int(input("Enter divisor: "))
    print(divident / divisor)
except ZeroDivisionError:
    print("Cannot divide by zero")
except ValueError:
    print("Invalid input")
except Exception as err:
    print("Unexpected Error:", err)
```

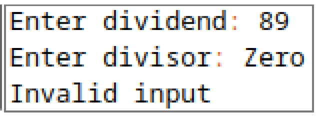
Output:



```
Enter dividend: 89
Enter divisor: 4
22.25
```



```
Enter dividend: 89
Enter divisor: 0
Cannot divide by zero
```



```
Enter dividend: 89
Enter divisor: Zero
Invalid input
```

Result:

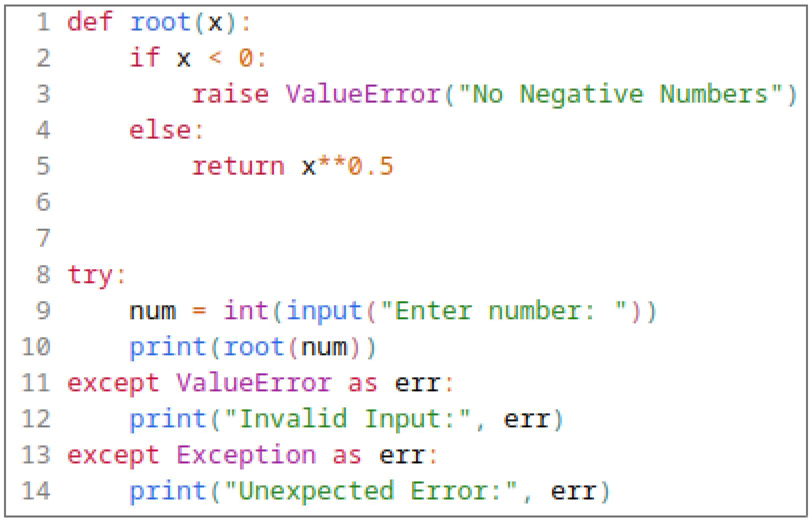
The Python Program to divide two numbers with multiple exception handlers has been successfully written and executed.

12.2 – Square root of number

Aim:

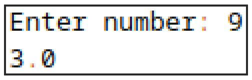
Write a Python Program to find the square root of a number with multiple exception handlers.

Source Code:

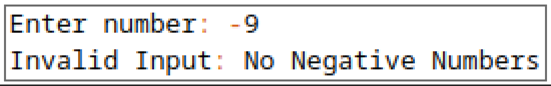


```python
def root(x):
    if x < 0:
        raise ValueError("No Negative Numbers")
    else:
        return x**0.5


try:
    num = int(input("Enter number: "))
    print(root(num))
except ValueError as err:
    print("Invalid Input:", err)
except Exception as err:
    print("Unexpected Error:", err)
```

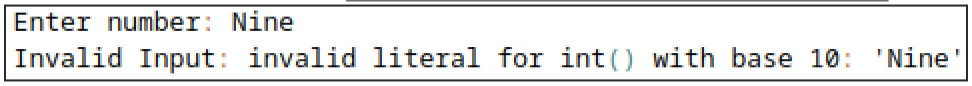
Output:



```
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:

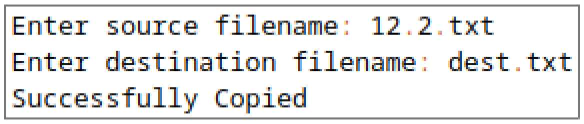
```python
try:
    filename = input("Enter source filename: ")

    with open(filename, "r") as f:
        data = f.read()

    filename = input("Enter destination filename: ")

    with open(filename, "w") as f:
        f.write(data)
except FileNotFoundError:
    print("Error: File not found.")
except Exception as e:
    print("Unexpected error:", e)
else:
    print("Successfully Copied")
```

Input:

12.2.txt:

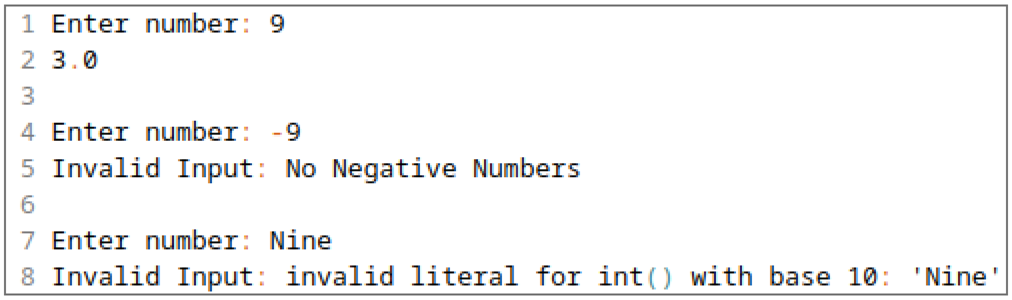
```
Enter number: 9
3.0

Enter number: -9
Invalid Input: No Negative Numbers

Enter number: Nine
Invalid Input: invalid literal for int() with base 10: 'Nine'
```

Output:



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

dest.txt:



```
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 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:

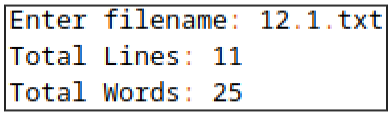
```python
try:
    filename = input("Enter filename: ")

    with open(filename, "r") as f:
        lines = f.readlines()

    line_count = len(lines)
    word_count = sum(len(line.split()) for line in lines)

    print("Total Lines:", line_count)
    print("Total Words:", word_count)

except FileNotFoundError:
    print("Error: File not found.")
except Exception as e:
    print("Unexpected error:", e)
```

Input:

12.1.txt:

```
Enter dividend: 89
Enter divisor: 4
22.25

Enter dividend: 89
Enter divisor: 0
Cannot divide by zero

Enter dividend: 89
Enter divisor: Zero
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.