

Module – 1

SIMPLE PROGRAMS

Aim: 1.1 Demonstrate functions and all categories of functions in C/C++/java/python.

Code: C: Function

```
#include <stdio.h>

int main()
{
    printf("Hello!\n");
    return 0;
}
```

C++: Function

```
#include <iostream>
using namespace std;
int add(int x, int y)
{
    return x + y;
}

int main() {
    int hello = add(4,5);
    cout<<hello;
    return 0;
}
```

Java: Function

```
class Main
{
```

```
public static void printSum(int a, int b)
{
    System.out.println("Sum: " + (a + b));
}
public static void main(String[] args) {
    printSum(4, 5);
}
}
```

Python: Function

```
def Sample():
    print( "Welcome!")
Sample()
```

Output: Output of C:

Output
Hello!

Output of C++:

Output
9

Output of Java:

Output
Sum: 9

Output of Python:

Output

Welcome!

Aim: 1.2 Demonstrate arrays (Insertion, Deletion, arithmetic operations).

Code:

```
arr = [10, 20, 30, 40]
print("Original Array:", arr)

arr.insert(2, 25)
print("After Insertion (25 at index 2):", arr)

del arr[3]
print("After Deletion (element at index 3 removed):", arr)

total = sum(arr)
print("Sum of elements:", total)

average = total / len(arr)
print("Average of elements:", average)

arr2 = [5, 10, 15, 20]
min_len = min(len(arr), len(arr2))
sum_array = [arr[i] + arr2[i] for i in range(min_len)]
print("Element-wise addition with", arr2, ":", sum_array)
```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5
Original Array: [10, 20, 30, 40]
After Insertion (25 at index 2): [10, 20, 25, 30, 40]
After Deletion (element at index 3 removed): [10, 20, 25, 40]
Sum of elements: 95
Average of elements: 23.75
Element-wise addition with [5, 10, 15, 20] : [15, 30, 40, 60]
○ PS D:\SEM 5\Python\IU2341230156\IU2341230156>
    
```

Aim: 1.3 Perform Addition, Multiplication for Vectors and Matrices.

Code:

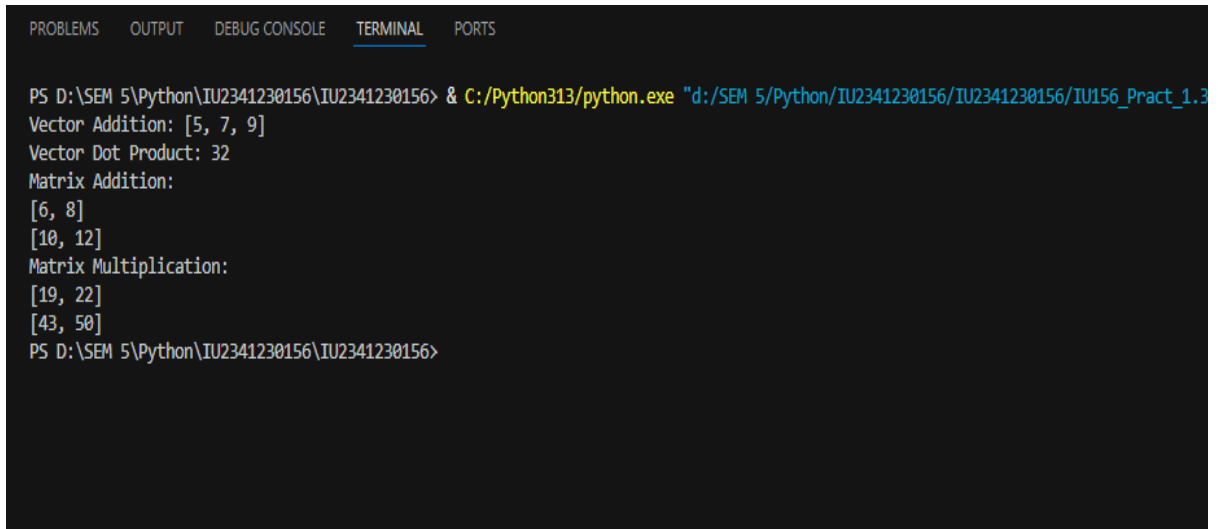
```

v1 = array('i', [1, 2, 3])
v2 = array('i', [4, 5, 6])
vector_add = array('i', [v1[i] + v2[i] for i in range(len(v1))])
vector_dot_product = sum(v1[i] * v2[i] for i in range(len(v1)))
print("Vector Addition:", list(vector_add))
print("Vector Dot Product:", vector_dot_product)
m1 = [array('i', [1, 2]), array('i', [3, 4])]
m2 = [array('i', [5, 6]), array('i', [7, 8])]
matrix_add = [array('i', [m1[i][j] + m2[i][j] for j in range(len(m1[0]))]) for i in
range(len(m1))]
matrix_mul = []
for i in range(len(m1)):
    row = array('i', [])
    for j in range(len(m2[0])):
        val = sum(m1[i][k] * m2[k][j] for k in range(len(m2)))
        row.append(val)
    matrix_mul.append(row)
print("Matrix Addition:")
for row in matrix_add:
    
```

```

print(list(row))
print("Matrix Multiplication:")
for row in matrix_mul:
    print(list(row))
    
```

Output:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/IU2341230156/Pract_1.3
Vector Addition: [5, 7, 9]
Vector Dot Product: 32
Matrix Addition:
[6, 8]
[10, 12]
Matrix Multiplication:
[19, 22]
[43, 50]
PS D:\SEM 5\Python\IU2341230156\IU2341230156>
    
```

Aim: 1.4 Practical based on Strings (Length finding, change specific character, palindrome, concatenation)

Code:

```

print("\n##### Length #####")
s = "hello"
print(f"The length of string {s}: {len(s)}")
print("\n##### Change Specific Character #####")
print("Replace l in 'hello' with s")
print(s.replace("l", "s"))
print("\n##### Concatenation #####")
print("The concatenation of s and n")
n = "world"
result = s + " " + n
print(result)
print("\n##### Palindrome #####")
    
```

```

h= "level"
print(f"Printing a palindrome for string: '{h}'")
print(h == h[::-1])
    
```

Output:

```

PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/IU2341230156

##### Length #####
The lenght of string hello: 5

##### Change Specific Character #####
Replace l in 'hello' with s
hesso

##### Concatenation #####
The concatenation of s and n
hello world

##### Palindrome #####
Printing a palindrome for 'level'
True
    
```

Aim: 1.5 Create a menu driven program to show various operators supported by python.

Code:

```

a = int(input("Enter number a:"))
b = int(input("Enter number b:"))
c = int(input("Enter the choice:\n1. Addition\n2. Multiplication\n3. Substaction\n4.
Division\n"))
    
```

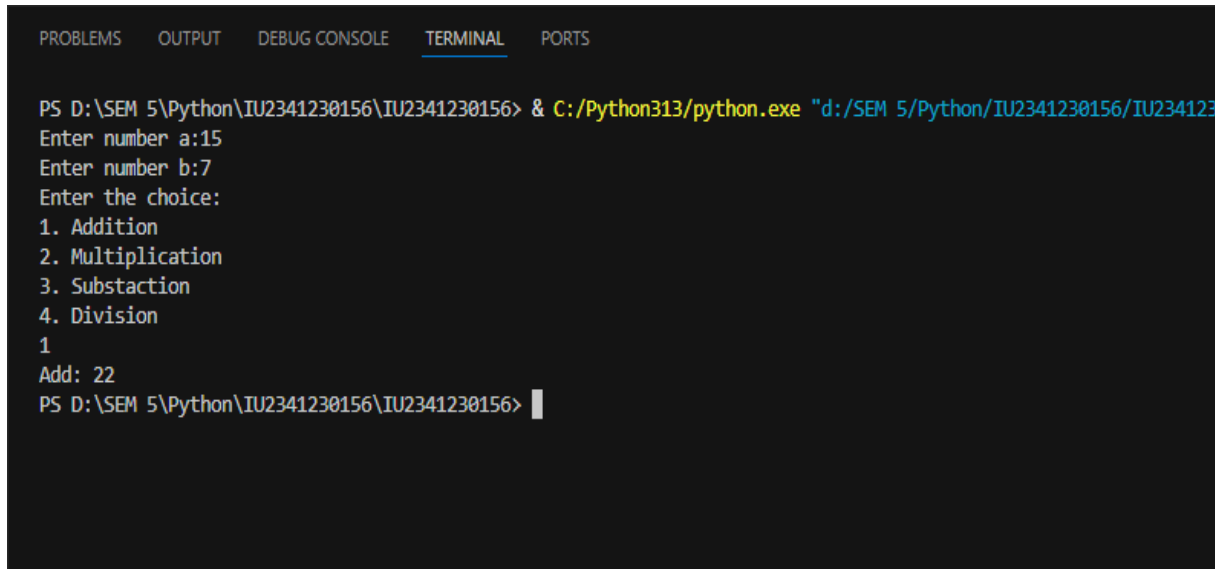
```

match c:
    case 1:
        print(f"Add: {a+b}")
    case 2:
        print(f"Mult: {a*b}")
    case 3:
        print(f"Sub: {a-b}")
    
```

```

case 4:
    print(f" Division: {a/b}")
case _:
    print("Enter proper number!")
    
```

Output:



```

PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/IU2341230156"
Enter number a:15
Enter number b:7
Enter the choice:
1. Addition
2. Multiplication
3. Substaction
4. Division
1
Add: 22
PS D:\SEM 5\Python\IU2341230156\IU2341230156>
    
```

Aim: 1.6 Write a python program to find out if a given number is even or odd using a user defined function.

Code:

```

n = int(input("Enter a number to check:"))
def is_even_odd(n):
    if n % 2 == 0:
        print(f"{n} is even.")
    else:
        print(f"{n} is odd.")

is_even_odd()
    
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/IU156_Pr
Enter a number to check:56
56 is even.
PS D:\SEM 5\Python\IU2341230156\IU2341230156> |
```

- Aim: 1.7**
- 1) Addition of first 15 numbers using loop.
 - 2) Addition of any 15 numbers using a loop.

Code: `### 1) Addition of first 15 numbers using loop.`

```
sum = 0
```

```
for i in range(1,16):
```

```
    sum += i
```

```
print(sum)
```

`### 2) Addition of any 15 numbers using a loop.`

```
sum1 = 0
```

```
for j in range(1,16):
```

```
    a = int(input(f"Enter {j} number: "))
```

```
    sum += a
```

```
print(f"The sum of numbers you entered: {sum}")
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/IU2341230156.py"
Enter 1 number: 3
Enter 2 number: 4
Enter 3 number: 5
Enter 4 number: 6
Enter 5 number: 7
Enter 6 number: 1
Enter 7 number: 4
Enter 8 number: 2
Enter 9 number: 09
Enter 10 number: 3
Enter 11 number: 5
Enter 12 number: 78
Enter 13 number: 3
Enter 14 number: 2
Enter 15 number: 7
The sum of numbers you entered: 259
PS D:\SEM 5\Python\IU2341230156\IU2341230156> █
```

Aim: 1.8 Write a python program to check if the entered year is leap year or not.

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

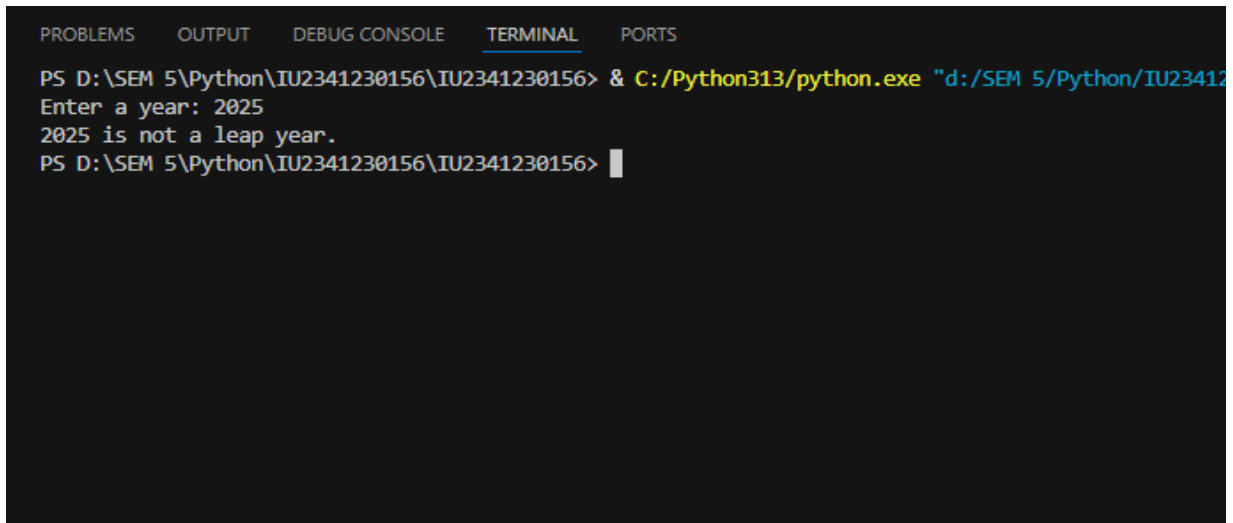
`if a % 4 == 0 or a % 100 == 0 and 400 != 0:`

`print(f"{a} is leap year.")`

`else:`

`print(f"{a} is not a leap year.")`

Output:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/leap_year.py"
Enter a year: 2025
2025 is not a leap year.
PS D:\SEM 5\Python\IU2341230156\IU2341230156>
```

Aim: 1.9 Write a Python program which accepts the user's first and last name and print them in reverse order with a space between them.

Code: `from operator import concat`
`first = input("Enter you first name: ")`
`last = input("Enter you last name: ")`
`correct_order = first + " " + last`
`print(correct_order)`
`rev_ = correct_order[::-1]`
`print(rev_)`

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/IU2341230156
Enter you first name: mayank
Enter you last name: raval
mayank raval
lavar knayam
PS D:\SEM 5\Python\IU2341230156\IU2341230156>
    
```

Aim: 1.10 Write a Python program to print the calendar of a given month and year.

Code:

```

import calendar
year = 2025
month = 7
print(calendar.month(year, month))
    
```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/IU2341230156
July 2025
Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
PS D:\SEM 5\Python\IU2341230156\IU2341230156>
    
```

Aim: 1.11 Write a Python program to check whether a specified value is contained in a group of values. Test Data: c. -> [1, 5, 8, 3]: True - 1 -> [1, 5, 8, 3]: False

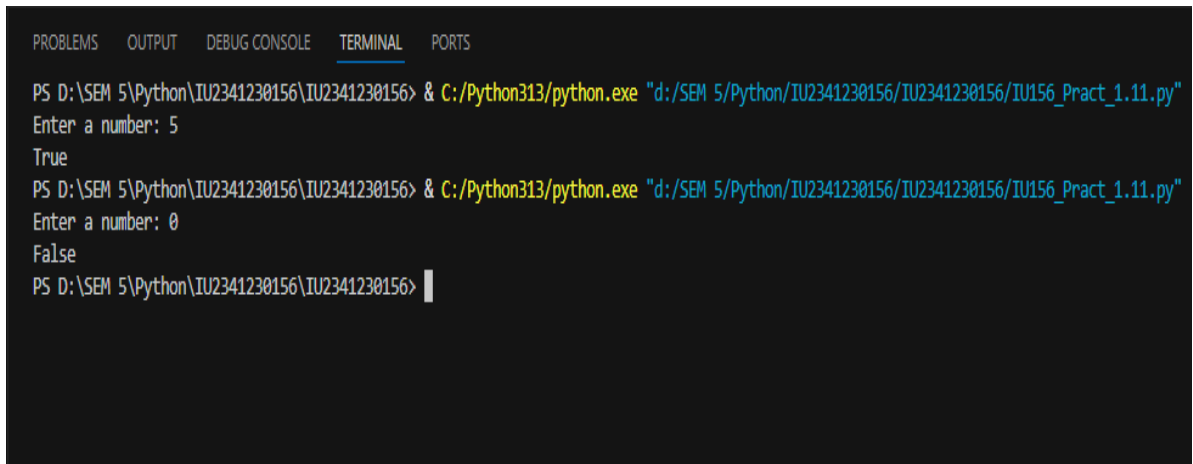
Code:

```

a = int(input("Enter a number: "))
test_data = [1, 5, 8, 3]
    
```

```
if a in test_data:  
    print("True")  
else:  
    print("False")
```

Output:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS D:\SEM 5\Python\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/IU156_Pract_1.11.py"  
Enter a number: 5  
True  
PS D:\SEM 5\Python\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/IU156_Pract_1.11.py"  
Enter a number: 0  
False  
PS D:\SEM 5\Python\IU2341230156> |
```

Aim: 1.12 Write a Python program to get OS name, platform and release information.

Code:

```
import platform  
uname_os = platform.uname()  
print(uname_os)  
print(uname_os.release)  
print(uname_os.version)
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/IU2341230156/IU156_Pract_1.12.py"
uname_result(system='Windows', node='Mayank', release='11', version='10.0.26100', machine='AMD64')
11
10.0.26100
PS D:\SEM 5\Python\IU2341230156\IU2341230156> |
```

Module - 2

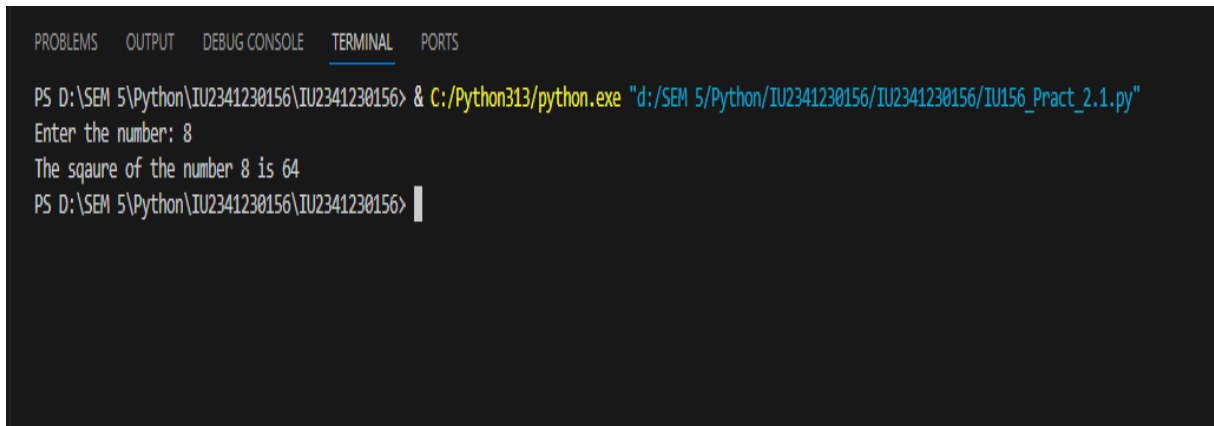
USER DEFINED FUNCTIONS (RECURSION, ARGUMENT PASSING, SCOPING)

Aim: 2.1 Write a function to find out x^y . Function should find out the square of x in case of default argument passing.

```
Code: x = int(input("Enter the number: "))  
def square_to(x: int, y: int = 2):  
  
    return x.__pow__(y)
```

```
print(f"The square of the number {x} is {square_to(x)}")
```

Output:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/IU156_Pract_2.1.py"  
Enter the number: 8  
The square of the number 8 is 64  
PS D:\SEM 5\Python\IU2341230156\IU2341230156> |
```

Aim: 2.2 Write a function to find out the factorial of a given number. I) without recursion II) with recursion

```
Code: def fact_without_recur(n):
```

```

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

```

num = 5
print("The 1st one:")
print(f"The factorial of {num} is: {fact_without_recur(num)}")
print("#####")
print("The 2nd one:")
    
```

```

def fact_with_recur(m):
    if m == 0 or m == 1:
        return 1
    return m * fact_with_recur(m - 1)

number = 5
result = fact_with_recur(number)
print(f"The factorial of {number} is {result}")
    
```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU23
The 1st one:
The factorial of 5 is: 120
#####
The 2nd one:
The factorial of 5 is 120
PS D:\SEM 5\Python\IU2341230156\IU2341230156> 
    
```

Aim: 2.3 Write a program to find out Fibonacci series using recursion and function as an object.

```

Code: def fibonacci_generator():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b

print("Fibonacci using function and generator:")
def print_fibonacci_series(n):
    fib_gen = fibonacci_generator()
    for _ in range(n):
        print(next(fib_gen), end=" ")
    print()

print_fibonacci_series(10)

def fibonacci_recursive(n):
    if n <= 1:
        return n
    else:
        return fibonacci_recursive(n-1) + fibonacci_recursive(n-2)

print("\n#####\n")
print("Fibonacci using recursive func:")
def print_fibonacci_recursive(n):
    for i in range(n):
        print(fibonacci_recursive(i), end=" ")
    print()

print_fibonacci_recursive(10)
  
```


Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/Fibonacci using function and generator.py"
0 1 1 2 3 5 8 13 21 34

#####

Fibonacci using recursive func:
0 1 1 2 3 5 8 13 21 34
PS D:\SEM 5\Python\IU2341230156\IU2341230156>
    
```

Aim: 2.4 4. Demonstrate Function Scoping.

Code:

```

bete = "deep"
print(f"global = {bete}")
def print_bete():
    global bete
    bete = "manek"
    print(f"inside function = {bete}")
print_bete()
print(f"after function = {bete}")
    
```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156/Function Scoping.py"
global = deep
inside function = manek
after function = manek
PS D:\SEM 5\Python\IU2341230156\IU2341230156>
    
```

Module - 3

HIGHER ORDER FUNCTIONS (LAMDA, MAP, FILTER)

Aim: 3.1 . Write a Python program to check whether a specified value is contained in a group of values using lambda function.

Test Data: 3 -> [1, 5, 8, 3] : True

-1 -> [1, 5, 8, 3] : False

Code:

```

test_data = [1, 5, 8, 3]
print("The test_data = [1, 5, 8, 3]")
num = int(input("Enter the number you want to check: "))
x = lambda test_data,num : True if num in test_data else False
if(x(test_data,num)):
    print(f"The element i.e {num} you have search for is present in the list")
else:
    print(f"The element i.e {num} you have search for is NOT present in the list")
    
```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2
The test_data = [1, 5, 8, 3]
Enter the number you want to check: 4
The element i.e 4 you have search for is NOT present in the list
PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2
The test_data = [1, 5, 8, 3]
Enter the number you want to check: 8
The element i.e 8 you have search for is present in the list
PS D:\SEM 5\Python\IU2341230156\IU2341230156>
    
```

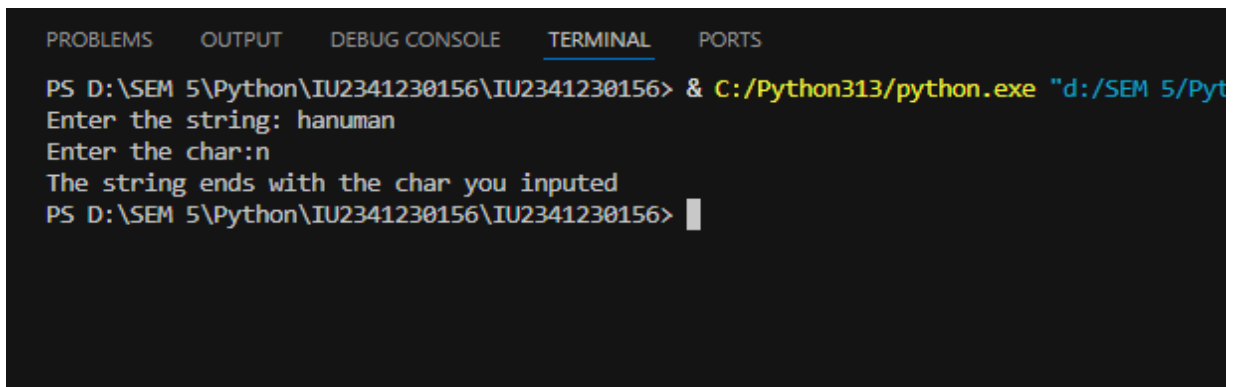
Aim: 3.2 2. Write a Python program to find whether a given string ends with a given character using Lambda.

Code:

```

a_string = input("Enter the string: ")
char=input("Enter the char:")
checker = a_string.endswith(char)
x = lambda checker : True if checker else False
if(x(checker)):
    print("The string ends with the char you inputed")
else:
    print("The string does not end with the char you inputed")
    
```

Output:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Pyt
Enter the string: hanuman
Enter the char:n
The string ends with the char you inputed
PS D:\SEM 5\Python\IU2341230156\IU2341230156>
    
```

Aim: 3.3 **3. Prime or not prime. Input: L= [3,4,6,9,11] Output: L= [P, NP, NP, NP, P] using map function.**

Code:

```

def is_prime(n):
    if n <= 1:
        return 'NP'
    if n <= 3:
        return 'P' # 2 and 3 are prime
    if n % 2 == 0 or n % 3 == 0:
        return 'NP' # Multiples of 2 or 3 (except 2 and 3 themselves) are not prime

    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return 'NP'
    
```

```

i += 6
return 'P'

```

```

L = [3, 4, 6, 9, 11]
print(f"First the List is: {L}")
prime_or_not = list(map(is_prime, L))
print(f"Prime or not in the list: {prime_or_not}")

```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5,
First the List is: [3, 4, 6, 9, 11]
Prime or not in the list: ['P', 'NP', 'NP', 'NP', 'P']
PS D:\SEM 5\Python\IU2341230156\IU2341230156>

```

Aim: 3.4 Write a python program to find out even numbers from a list using filter ().

Code:

```

def is_even(num):
    return num % 2 == 0
og_list = [1,2,3,4,5,6]
even_list = list(filter(is_even,og_list))
print(f"The original list is : {og_list}")
print(f"The even number from the list is : {even_list}")

```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM
The original list is : [1, 2, 3, 4, 5, 6]
The even number from the list is : [2, 4, 6]
PS D:\SEM 5\Python\IU2341230156\IU2341230156>

```

Module - 4

Importing of modules

Aim: 4 An interactive program where one module asks numbers from the user and the second module performs at least three arithmetic operations on them.

Code:

```
##### user_input_module #####
a = int(input("Enter the number a: "))
b = int(input("Enter the number b: "))

##### main.py #####
from user_input_module import a,b

add = a + b
mul = a * b
div = a / b
sub = a - b
print(f"Addition: {add}\nSubtraction: {sub}\nMultiplication: {mul}\nDivision: {div}")
```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Pyth
Enter the number a: 5
Enter the number b: 6
Addition: 11
Subtraction: -1
Multiplication: 30
Division: 0.8333333333333334
PS D:\SEM 5\Python\IU2341230156\IU2341230156>
    
```

Module - 5

FILE HANDLING

Aim: 5.1 A program to count the number of words, number of lines, occurrence of particular word, occurrence of particular character, number of blank spaces in a text file.

Code: ##### File Operations:

```
def analyze_text_file(filename, target_word, target_char):  
    num_lines = 0  
    num_words = 0  
    word_occurrence = 0  
    char_occurrence = 0  
    num_blank_spaces = 0  
  
    with open(filename, 'r') as file:  
        for line in file:  
            num_lines += 1  
            num_blank_spaces += line.count(' ')  
  
            words = line.split()  
            num_words += len(words)  
  
            # occurrence of particular word  
            for word in words:  
                if word.lower() == target_word.lower():  
                    word_occurrence += 1  
  
            # occurrence of particular character  
            char_occurrence += line.count(target_char)  
    results = {  
        "number_of_words": num_words,
```

```

"number_of_lines": num_lines,
f"occurrence_of_{target_word}": word_occurrence,
f"occurrence_of_{target_char}": char_occurrence,
"number_of_blank_spaces": num_blank_spaces
}
return results

```

```

file_to_analyze = "mayank.txt"
word_to_find = "example"
char_to_find = "k"

```

```

with open(file_to_analyze, 'w') as f:
    f.write("This is an example text file.\n")
    f.write("It contains multiple lines and words.\n")
    f.write("Let's see how many 'example' words and 'e' characters are present.\n")
    f.write(" This line has leading spaces and an extra space.\n")

```

```

analysis_results = analyze_text_file(file_to_analyze, word_to_find, char_to_find)

```

```

if analysis_results:
    for key, value in analysis_results.items():
        print(f"{key.replace('_', ' ').capitalize()}: {value}")

```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156
● Number of words: 32
  Number of lines: 4
  Occurrence of example: 1
  Occurrence of k: 0
  Number of blank spaces: 30
○ PS D:\SEM 5\Python\IU2341230156\IU2341230156>

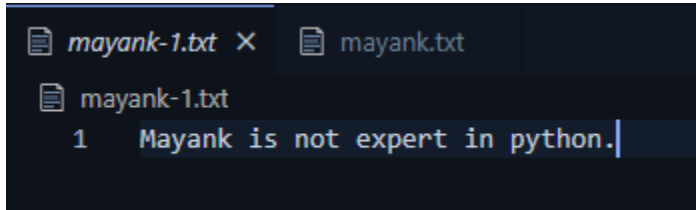
```

Aim: 5.2 A program to read a string from the user and append it into a file.

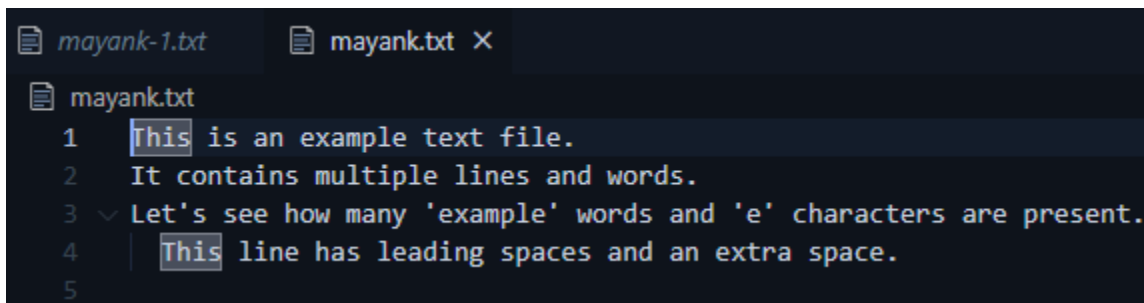
Code: ##### A program to read a string from the user and append it into a file.

```
filename = "mayank-1.txt"
str = "Mayank is not expert in python."
with open(filename, 'w') as file:
    for i in str:
        file.write(i)
```

Output:



```
mayank-1.txt X  mayank.txt
mayank-1.txt
1  Mayank is not expert in python.
```



```
mayank-1.txt  mayank.txt X
mayank.txt
1  This is an example text file.
2  It contains multiple lines and words.
3  Let's see how many 'example' words and 'e' characters are present.
4  This line has leading spaces and an extra space.
5
```

Aim: 5.3 A program to copy the contents of one file into another.

Code: ### A program to copy the contents of one file into another.

```
file1 = "mayank.txt"
file2 = "mayank-copy.txt"

with open(file1, 'r') as reader:
    with open(file2, 'w') as file:
        file.write(reader.read())
    print("Done!!")
```


Output:

```

mayank-1.txt X  mayank.txt
mayank-1.txt
1 Mayank is not expert in python.

```

```

mayank-copy.txt X
mayank-copy.txt
1 This is an example text file.
2 It contains multiple lines and words.
3 Let's see how many 'example' words and 'e' characters are present.
4 This line has leading spaces and an extra space.
5

```

Aim: 5.4 A program to read a text file and print all the numbers present in the text file.

Code:

```

### A program to read a text file and print all the numbers present in the text file.
filename = "mayank.txt"
with open(filename, 'r') as primary:
    all_content = primary.read()
    numbers = []
    for i in all_content:
        if i.isdigit():
            numbers.append(int(i))
    print(numbers)

```

Output:

```

mayank.txt X
mayank.txt
1 This is1 an example text file.
2 It contains2 multiple lines and words.
3 Let's see how many 2'example' words and 'e' characters are present.
4 This 1783ine has leading spaces and an extra space.
5

```

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● PS D:\SEM 5\Python\IU2341230156\IU2341230156> & C:/Python313/python.exe "d:/SEM 5/Python/IU2341230156
[1, 2, 2, 7, 8, 3]
● PS D:\SEM 5\Python\IU2341230156\IU2341230156>
    
```

Aim: 5.5 A program to append the contents of one file to another file.

Code: ##### A program to append the contents of one file to another file.

```
file1 = "mayank.txt"
```

```
file2 = "mayank-copy.txt"
```

```
with open(file1, 'r') as reader:
```

```
with open(file2,'w') as file:
```

```
file.write(reader.read())
```

```
print("Done!!")
```

Output:

```

mayank-1.txt X  mayank.txt
mayank-1.txt
1  Mayank is not expert in python.
    
```

```

mayank-copy.txt X
mayank-copy.txt
1  This is an example text file.
2  It contains multiple lines and words.
3  Let's see how many 'example' words and 'e' characters are present.
4  This line has leading spaces and an extra space.
5
    
```

Aim: 5.6 A program to read a file and capitalize the first letter of every word in the file.

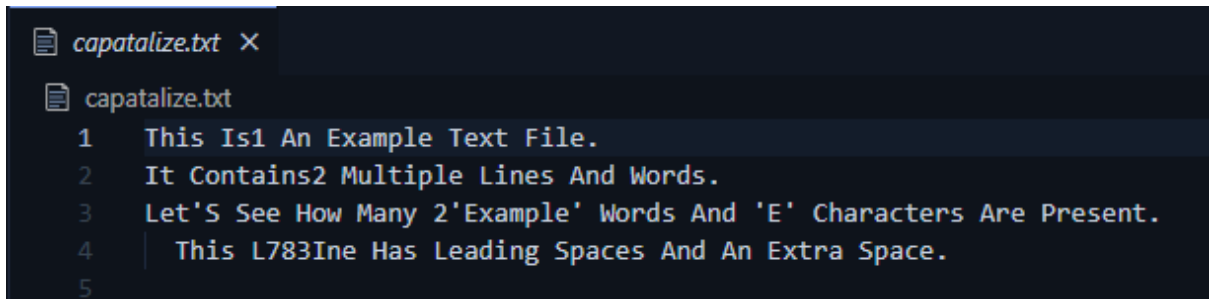
Code: ##### A program to read a file and capitalize the first letter of every word in the file.

```
filename = "mayank.txt"
```

```
with open(filename, 'r') as infile:
```

```
content = infile.read()
capitalized_content = content.title()
outfile = "capatalize.txt"
with open(outfile, 'w') as outfile:
    outfile.write(capitalized_content)
```

Output:



```
capatalize.txt X
capatalize.txt
1 This Is1 An Example Text File.
2 It Contains2 Multiple Lines And Words.
3 Let'S See How Many 2'Example' Words And 'E' Characters Are Present.
4   This L783Ine Has Leading Spaces And An Extra Space.
5
```

Module - 6

Collection Object Programs

Aim: 6.1 Write a Python program which accepts a sequence of comma-separated numbers from the user and generate a list and a tuple with those numbers.

Code:

```
sentence = input("Enter a sequence of comma-separated numbers: ")
numbers_list = [x.strip() for x in sentence.split(',')]
numbers_tuple = tuple(numbers_list)
print("List:", numbers_list)
print("Tuple:", numbers_tuple)
```

Output:

```
Enter a sequence of comma-separated numbers: may,anf,fe8h
List: ['may', 'anf', 'fe8hb']
Tuple: ('may', 'anf', 'fe8hb')
```

Aim: 6.2 Write a Python program to display the first and last colours from the following

Code:

```
colors = ["Red","Green","White" ,"Black"]

print("First color:", colors[0])
print("Last color:", colors[-1])
```

Output:

```
→ First color: Red
   Last color: Black
```

Aim: 6.3 Write a Python program to concatenate all elements in a list into a string and return it.

Code:

```
my_list = [1, 2, 3, 4, 5, " Printed 😊"]
second = ' '.join(map(str, my_list))
print(second)
```

Output:



Aim: 6.4 Write a Python program to print out a set containing all the colours from color_list_1 which are not present in color_list_2. Test Data: color_list_1 = set(["White", "Black", "Red"]), color_list_2 = set(["Red", "Green"]) Expected Output: {'Black', 'White'}

Code:

```
color_list_1 = set(["White", "Black", "Red"])
color_list_2 = set(["Red", "Green"])
```

```
diff = color_list_1 - color_list_2
sum = color_list_1 | color_list_2
```

```
print(diff)
print(sum)
```

Output:



Aim: 6.5 Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are squares of keys. Sample Dictionary: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10:100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225}

Code:

```
square_dict = {}
```

```
for i in range(1, 16):
```

```
square_dict[i] = i**2
```

```
print(square_dict)
```

Output:

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225}
```

Aim: 6.6 Write a menu driven program to implement the following methods on List.
(1) Create (2) Update particular element of list (3) Append to the list (4) Delete whole list (5) Delete particular element (6) Sort the list (7) Find length

Code: my_list = []

```

while True:
    print("\n----- MENU -----")
    print("1. Create a new list")
    print("2. Update a particular element of the list")
    print("3. Append an element to the list")
    print("4. Delete the whole list")
    print("5. Delete a particular element from the list")
    print("6. Sort the list")
    print("7. Find the length of the list")
    print("8. Exit")
    print("-----")
    choice = int(input("Enter your choice (1-8): "))
    match choice:
        case 1:
            my_list = []
            num_elements = int(input("Enter the number of elements to add to the new list: "))
            for i in range(num_elements):
                element = input(f"Enter element {i + 1}: ")

```

```
my_list.append(element)
print("List created successfully:", my_list)
```

case 2:

```
if not my_list:
    print("The list is empty. Please create a list first.")
else:
    index = int(input("Enter the index of the element to update: "))
    if 0 <= index < len(my_list):
        new_element = input("Enter the new element: ")
        my_list[index] = new_element
        print("Element updated successfully:", my_list)
    else:
        print("Invalid index. Please enter an index within the range of the list.")
```

case 3:

```
element_to_append = input("Enter the element to append: ")
my_list.append(element_to_append)
print("Element appended successfully:", my_list)
```

case 4:

```
my_list.clear()
print("List deleted successfully.")
```

case 5:

```
if not my_list:
    print("The list is empty. Nothing to delete.")
else:
    element_to_delete = input("Enter the element to delete: ")
    if element_to_delete in my_list:
        my_list.remove(element_to_delete)
        print("Element deleted successfully:", my_list)
```

```
else:  
    print("Element not found in the list.")  
case 6:  
    if not my_list:  
        print("The list is empty. Nothing to sort.")  
    else:  
        my_list.sort()  
    print("List sorted successfully:", my_list)  
  
case 7:  
    print("The length of the list is:", len(my_list))  
  
case 8:  
    print("Exiting the program. Goodbye!")  
    break
```

Output:

```
➡  
----- MENU -----  
1. Create a new list  
2. Update a particular element of the list  
3. Append an element to the list  
4. Delete the whole list  
5. Delete a particular element from the list  
6. Sort the list  
7. Find the length of the list  
8. Exit  
-----  
Enter your choice (1-8): 1  
Enter the number of elements to add to the new list: 2  
Enter element 1: 7  
Enter element 2: 4  
List created successfully: ['7', '4']  
  
----- MENU -----  
1. Create a new list  
2. Update a particular element of the list  
3. Append an element to the list  
4. Delete the whole list  
5. Delete a particular element from the list  
6. Sort the list  
7. Find the length of the list  
8. Exit  
-----  
Enter your choice (1-8): 8  
Exiting the program. Goodbye!
```


Aim: 6.7 Write a menu driven program to implement the following methods on set. (1) Create (2) Update particular element of set (3) Append to the set (4) Delete whole set (5) Delete particular element (6) Sort the set (7) Find length

```
Code: def display_menu():
    print("\nSet Operations Menu:")
    print("1. Create Set")
    print("2. Update Set with Multiple Elements")
    print("3. Append (Add) Single Element")
    print("4. Delete Whole Set")
    print("5. Delete Particular Element (by Value)")
    print("6. Sort Set (converted to sorted list)")
    print("7. Find Length of the Set")
    print("8. Exit")
    my_set = set()

    while True:
        display_menu()
        choice = input("Enter your choice (1-8): ")

        if choice == '1':
            elements = input("Enter elements separated by space: ").split()
            my_set = set(elements)
            print("Set created:", my_set)

            elif choice == '2':
                elements = input("Enter elements to update (space separated): ").split()
                my_set.update(elements)
                print("Updated Set:", my_set)

            elif choice == '3':
```

```
element = input("Enter element to add: ")
my_set.add(element)
print("Updated Set:", my_set)

elif choice == '4':
    my_set.clear()
    print("Set deleted. Current Set:", my_set)

elif choice == '5':
    element = input("Enter element to remove: ")
    if element in my_set:
        my_set.remove(element)
        print("Updated Set:", my_set)
    else:
        print("Element not found!")

elif choice == '6':
    sorted_list = sorted(my_set)
    print("Sorted Set (as list):", sorted_list)

elif choice == '7':
    print("Length of the Set:", len(my_set))

elif choice == '8':
    print("Exiting program.")
    break

else:
    print("Invalid choice! Please select between 1 and 8.")
```

Output:

```

Set Operations Menu:
1. Create Set
2. Update Set with Multiple Elements
3. Append (Add) Single Element
4. Delete Whole Set
5. Delete Particular Element (by Value)
6. Sort Set (converted to sorted list)
7. Find Length of the Set
8. Exit
Enter your choice (1-8): 1
Enter elements separated by space: 3
Set created: {'3'}

Set Operations Menu:
1. Create Set
2. Update Set with Multiple Elements
3. Append (Add) Single Element
4. Delete Whole Set
5. Delete Particular Element (by Value)
6. Sort Set (converted to sorted list)
7. Find Length of the Set
8. Exit
Enter your choice (1-8): 2
Enter elements to update (space separated): 3
Updated Set: {'3'}

Set Operations Menu:
1. Create Set
2. Update Set with Multiple Elements
3. Append (Add) Single Element
4. Delete Whole Set
5. Delete Particular Element (by Value)
6. Sort Set (converted to sorted list)
7. Find Length of the Set
8. Exit
Enter your choice (1-8): 3 3 54
Invalid choice! Please select between 1 and 8.
    
```

Aim: 6.8 Write a menu driven program to implement following methods on Dictionary.

(1) Create (2) Update particular element of dictionary (3) Append to the dictionary (4) Delete whole dictionary (5) Delete particular element (6) Sort the dictionary (7) Find length

Code:

```
def pract_6_8():
    my_dict = {}
    while True:
        print("\n--- Dictionary Operations Menu ---")
        print("1. Create Dictionary")
        print("2. Update Particular Element")
        print("3. Append to Dictionary")
        print("4. Delete Whole Dictionary")
        print("5. Delete Particular Element")
        print("6. Sort the Dictionary by Key")
        print("7. Find Length of Dictionary")
        print("8. Display Dictionary")
        print("9. Exit")
        choice = input("Enter your choice (1-9): ")
        if choice == '1':
            n = int(input("Enter number of elements: "))
            my_dict = {}
            for _ in range(n):
                key = input("Enter key: ")
                value = input("Enter value: ")
                my_dict[key] = value
            print("Dictionary created successfully.")
        elif choice == '2':
            key = input("Enter key to update: ")
            if key in my_dict:
                new_value = input("Enter new value: ")
                my_dict[key] = new_value
            print("Element updated.")
        else:
            print("Key not found.")
        elif choice == '3':
```

```
key = input("Enter key to append: ")
if key in my_dict:
    print("Key already exists. Use update option instead.")
else:
    value = input("Enter value: ")
    my_dict[key] = value
    print("Element appended.")
elif choice == '4':
    my_dict.clear()
    print("Dictionary deleted (cleared).")
elif choice == '5':
    key = input("Enter key to delete: ")
    if key in my_dict:
        del my_dict[key]
        print("Element deleted.")
    else:
        print("Key not found.")
elif choice == '6':
    if my_dict:
        sorted_dict = dict(sorted(my_dict.items()))
        print("Sorted Dictionary (by keys):", sorted_dict)
    else:
        print("Dictionary is empty.")
elif choice == '7':
    print("Length of Dictionary:", len(my_dict))
elif choice == '8':
    print("Current Dictionary:", my_dict)
elif choice == '9':
    print("Exiting Program.")
    break
else:
```

```
print("Invalid choice. Please try again.")
```

```
pract_6_8()
```

Output:

```

...
--- Dictionary Operations Menu ---
1. Create Dictionary
2. Update Particular Element
3. Append to Dictionary
4. Delete Whole Dictionary
5. Delete Particular Element
6. Sort the Dictionary by Key
7. Find Length of Dictionary
8. Display Dictionary
9. Exit
Enter your choice (1-9): 1
Enter number of elements: 2
Enter key: name
Enter value: mayank
Enter key: age
Enter value: 20
Dictionary created successfully.

--- Dictionary Operations Menu ---
1. Create Dictionary
2. Update Particular Element
3. Append to Dictionary
4. Delete Whole Dictionary
5. Delete Particular Element
6. Sort the Dictionary by Key
7. Find Length of Dictionary
8. Display Dictionary
9. Exit
Enter your choice (1-9): 8
Current Dictionary: {'name': 'mayank', 'age': '20'}
```

Aim: 6.9 Do following:

- 1) Define and Call user defined function for n numbers one by one.
- 2) Check if each element is Even or Odd and Print it. (Use Class Variable)
- 3) Print List of even numbers and odd numbers.

Code:

```
def check_even_odd(numbers):
    even_numbers = []
    odd_numbers = []
```

```

for num in numbers:
    if num % 2 == 0:
        print(f"{num} is Even")
        even_numbers.append(num)
    else:
        print(f"{num} is Odd")
        odd_numbers.append(num)
print("\nEven Numbers List:", even_numbers)
print("Odd Numbers List:", odd_numbers)
def main():
    n = int(input("How many numbers do you want to check? "))
    numbers = []
    for i in range(n):
        num = int(input(f"Enter number {i+1}: "))
        numbers.append(num)
    check_even_odd(numbers)
main()
    
```

Output:

```

➡ How many numbers do you want to check? 3
Enter number 1: 12
Enter number 2: 13
Enter number 3: 14
12 is Even
13 is Odd
14 is Even

Even Numbers List: [12, 14]
Odd Numbers List: [13]
    
```

Aim: 6.10 Print total number of Even numbers and Total number of Odd Numbers.

Example:

1) function(21), function(22),function(35),function(36),function(40).

2) 21 is odd, 22 is even, 35 is odd,36 is even, 40 is even.

3) Even={22,36,40} Odd={21,35}

total number of even = 3, total number of odd=2

Code:

```
even_numbers = []
odd_numbers = []
def function(num):
    if num % 2 == 0:
        print(f"{num} is even")
        even_numbers.append(num)
    else:
        print(f"{num} is odd")
        odd_numbers.append(num)
function(21)
function(22)
function(35)
function(36)
function(40)
print(f"\nEven = {even_numbers}")
print(f"Odd = {odd_numbers}")
print(f"Total number of even = {len(even_numbers)}")
print(f"Total number of odd = {len(odd_numbers)}")
```

Output:

```
21 is odd
22 is even
35 is odd
36 is even
40 is even

Even = [22, 36, 40]
Odd = [21, 35]
Total number of even = 3
Total number of odd = 2
```


Module - 13

Numpy

Aim: 13.1 Extract all odd numbers from the array using “where” clause.

Code:

```
import numpy as np

arr = np.array([10, 21, 32, 43, 54, 65, 76])
odd_indices = np.where(arr % 2 != 0)
odd_numbers = arr[odd_indices]

print("Original Array:", arr)
print("Odd Numbers:", odd_numbers)
```

Output:

```
➔ Original Array: [10 21 32 43 54 65 76]
   Odd Numbers: [21 43 65]
```

Aim: 13.2 Replace all odd numbers in array with -1

Code:

```
import numpy as np

arr = np.array([10, 21, 32, 43, 54, 65, 76])
arr[arr % 2 != 0] = -1

print("Modified Array:", arr)
```

Output:


```
➔ Modified Array: [10 -1 32 -1 54 -1 76]
```

Aim: 13.3 Convert a 1D array to a 2D array with 2 rows and 5 columns.

Code:

```
import numpy as np
arr_1d = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
arr_2d = arr_1d.reshape(2, 5)
print("2D Array:\n", arr_2d)
```

Output:



```

2D Array:
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
    
```

Aim: 13.4 Get the common items between array1 and array2.

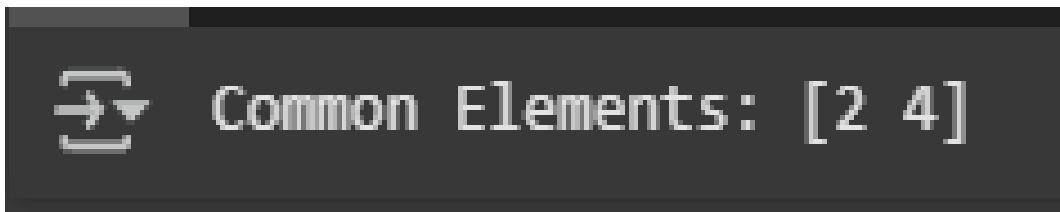
Input: array1 = [1,2,3,2,3,4,3,4,5,6],array2 = [7,2,10,2,7,4,9,4,9,8]

Desired Output: array([2, 4])

Code:

```
import numpy as np
array1 = np.array([1, 2, 3, 2, 3, 4, 3, 4, 5, 6])
array2 = np.array([7, 2, 10, 2, 7, 4, 9, 4, 9, 8])
common_elements = np.intersect1d(array1, array2)
print("Common Elements:", common_elements)
```

Output:



```

Common Elements: [2 4]
    
```

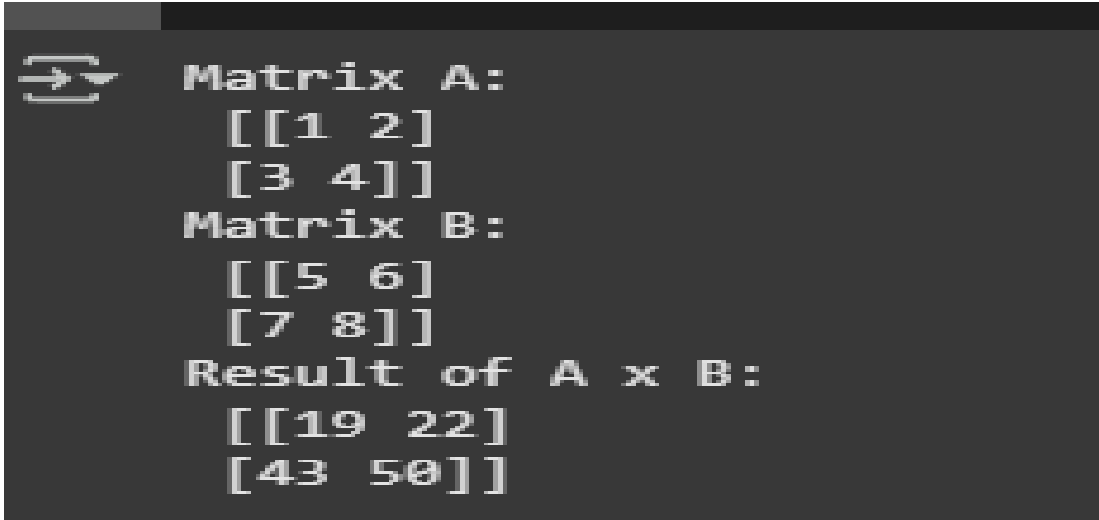
Aim: 13.5 Perform Matrix multiplication on 2 matrices.

Code:

```
import numpy as np
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
result = np.dot(A, B)
print("Matrix A:\n", A)
```

```
print("Matrix B:\n", B)
print("Result of A x B:\n", result)
```

Output:



```

Matrix A:
[[1 2]
 [3 4]]
Matrix B:
[[5 6]
 [7 8]]
Result of A x B:
[[19 22]
 [43 50]]
    
```

Aim: 13.6 Using numpy and matplotlib/pylab generate bar plots for appropriate data.

Code:

```

import numpy as np
import matplotlib.pyplot as plt

students = np.array(['Mayank', 'Maru', 'Jaydeep', 'Jhaveri', 'Goti'])
scores = np.array([85, 92, 78, 90, 88])

plt.bar(students, scores, color='skyblue')
plt.title("Student Scores")
plt.xlabel("Students")
plt.ylabel("Scores")
plt.show()
    
```

Output:

