Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

Experiment No:7

Aim:Write programs to demonstrate user-defined functions, lambda functions, and recursion.

#User-defined functions

1.Write a user-defined function to find the largest of three numbers.

```
a=int(input("Enter a:"))
b=int(input("Enter b:"))
c=int(input("Enter c:"))
def largest(a, b, c):
 return max(a, b, c)
print("Largest:",largest(a,b,c))
```

```
Enter a:4
Enter b:5
Enter c:3
Largest: 5
```

2.Create a function to calculate the factorial of a given number.

```
n=int(input("Enter n:"))
def factorial(n):
    fact = 1
    for i in range(1, n + 1):
        fact *= i
    return fact
print("Factorial:",factorial(n))
```

```
Enter n:5
Factorial: 120

=== Code Execution Successful ===
```

3. Define a function to check whether a given number is prime or not.

```
n=int(input("Enter n:"))
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True
print("Prime:",is_prime(n))
```

```
Enter n:5
Prime: True

=== Code Execution Successful ===
```

4. Write a function that takes a string as input and returns the number of vowels.

```
def count_vowels(s):
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

```
    vowels = "aeiouAEIOU"
    count = 0
    for ch in s:
        if ch in vowels:
            count += 1
    return count

# take user input
text = input("Enter a string: ")
print("Number of vowels in the string:", count_vowels(text))
```

```
Enter a string: Rural
Number of vowels in the string: 2

=== Code Execution Successful ===
```

5. Create a function that converts temperature from Celsius to Fahrenheit.

```
def celsius_to_fahrenheit(celsius):
    fahrenheit = (celsius * 9/5) + 32
    return fahrenheit

# take user input
c = float(input("Enter temperature in Celsius: "))
print("Temperature in Fahrenheit:", celsius_to_fahrenheit(c))
```

```
Enter temperature in Celsius: 37
Temperature in Fahrenheit: 98.6

=== Code Execution Successful ===
```

6. Define a function to find the sum of all even numbers in a list.

```
def sum_of_even_numbers(numbers):
    total = 0
    for num in numbers:
        if num % 2 == 0:
            total += num
    return total

# take user input (without map)
nums_input = input("Enter numbers separated by spaces: ")
nums = []  # empty list

for n in nums_input.split():
    nums.append(int(n))  # convert each string to int and add to list

print("Sum of all even numbers:", sum_of_even_numbers(nums))
```

```
Enter numbers separated by spaces: 1 2 3 4 5 6
Sum of all even numbers: 12

=== Code Execution Successful ===
```

7. Write a function that reverses a given string.

```
def reverse_string(s):
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

```
    return s[::-1]   # slicing method to reverse the string

# take user input
text = input("Enter a string: ")
print("Reversed string:", reverse_string(text))
```

```
Enter a string: text
Reversed string: txet

=== Code Execution Successful ===
```

8. Create a function that checks whether a string is a palindrome.

```
def is_palindrome(s):
    s = s.lower()  # make it case-insensitive
    reversed_s = s[::-1]
    return s == reversed_s

# take user input
text = input("Enter a string: ")

if is_palindrome(text):
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

```
Enter a string: noon
The string is a palindrome.

=== Code Execution Successful ===
```

9. Define a function that returns the maximum and minimum values from a list.

```
def find_max_min(numbers):
    maximum = max(numbers)
    minimum = min(numbers)
    return maximum, minimum

# take user input
nums_input = input("Enter numbers separated by spaces: ")
numbers = []

for n in nums_input.split():
    numbers.append(int(n))  # convert each to integer

max_val, min_val = find_max_min(numbers)
print("Maximum value:", max_val)
print("Minimum value:", min_val)
```

```
Enter numbers separated by spaces: 1 2 3 4 5 9
Maximum value: 9
Minimum value: 1

=== Code Execution Successful ===
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

10. Write a function to count the number of words in a sentence.

```
def count_words(sentence):
    words = sentence.split()   # split sentence by spaces
    return len(words)

# take user input
text = input("Enter a sentence: ")
print("Number of words in the sentence:", count_words(text))
```

```
Enter a sentence: Who are you ?
Number of words in the sentence: 4

=== Code Execution Successful ===
```

11. Create a function that accepts marks in 5 subjects and returns the percentage.

```
def calculate_percentage(marks):
    total = sum(marks)
    percentage = total / len(marks)
    return percentage

# take user input
marks = []

print("Enter marks for 5 subjects:")
for i in range(5):
    m = float(input(f"Subject {i+1}: "))
    marks.append(m)

print("Percentage:", calculate_percentage(marks), "%")
```

```
Enter marks for 5 subjects:
Subject 1: 95
Subject 2: 90
Subject 3: 95
Subject 4: 99
Subject 5: 98
Percentage: 95.4 %

=== Code Execution Successful ===
```

12. Define a function to calculate the area of a circle, rectangle, and triangle (use parameters).

```
def area_circle(radius):
    return 3.1416 * radius * radius

def area_rectangle(length, breadth):
    return length * breadth

def area_triangle(base, height):
    return 0.5 * base * height
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

```
# take user input
print("Choose the shape to calculate area:")
print("1. Circle")
print("2. Rectangle")
print("3. Triangle")

choice = int(input("Enter your choice (1/2/3): "))

if choice == 1:
    r = float(input("Enter the radius of the circle: "))
    print("Area of Circle =", area_circle(r))

elif choice == 2:
    l = float(input("Enter the length of the rectangle: "))
    b = float(input("Enter the breadth of the rectangle: "))
    print("Area of Rectangle =", area_rectangle(l, b))

elif choice == 3:
    base = float(input("Enter the base of the triangle: "))
    height = float(input("Enter the height of the triangle: "))
    print("Area of Triangle =", area_triangle(base, height))

else:
    print("Invalid choice!")
```

```
Choose the shape to calculate area:
1. Circle
2. Rectangle
3. Triangle
Enter your choice (1/2/3): 2
Enter the length of the rectangle: 3
Enter the breadth of the rectangle: 6
Area of Rectangle = 18.0

=== Code Execution Successful ===
```

13.Write a function that takes a list and returns a new list with only unique elements.

```
def unique_elements(lst):
    unique_list = []
    for item in lst:
        if item not in unique_list:
            unique_list.append(item)
    return unique_list

# take user input
nums_input = input("Enter numbers separated by spaces: ")
numbers = []

for n in nums_input.split():
    numbers.append(int(n))  # convert each input to integer

print("List with unique elements:", unique_elements(numbers))
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

```
Enter numbers separated by spaces: 3 3 4 5 6 7
List with unique elements: [3, 4, 5, 6, 7]

=== Code Execution Successful ===
```

14. Create a function that accepts a number and prints its multiplication table.

```
def multiplication_table(n):
    print(f"\nMultiplication Table of {n}:")
    for i in range(1, 11):
        print(f"{n} x {i} = {n * i}")

# take user input
num = int(input("Enter a number: "))
multiplication_table(num)
```

```
Enter a number: 5

Multiplication Table of 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

=== Code Execution Successful ===
```

15. Define a function that takes two lists and returns their element-wise sum.

```
def element_wise_sum(list1, list2):
    result = []
    for i in range(len(list1)):
        result.append(list1[i] + list2[i])
    return result

# take user input
print("Enter elements for the first list (space-separated):")
list1_input = input()
list1 = []
for n in list1_input.split():
    list1.append(int(n))

print("Enter elements for the second list (space-separated):")
list2_input = input()
list2 = []
for n in list2_input.split():
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

```
    list2.append(int(n))

# check if both lists are of equal length
if len(list1) != len(list2):
    print("Error: Both lists must have the same number of elements.")
else:
    print("Element-wise sum:", element_wise_sum(list1, list2))
```

```
Enter elements for the first list (space-separated):
1 2 3 4 5
Enter elements for the second list (space-separated):
6 7 8 9 0
Element-wise sum: [7, 9, 11, 13, 5]

=== Code Execution Successful ===
```

#Lambda functions

1.Write a lambda function to add two numbers

```
# lambda function to add two numbers
add = lambda a, b: a + b

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

print("Sum of the two numbers:", add(a, b))
```

```
Enter first number: 5
Enter second number: 3
Sum of the two numbers: 8.0

=== Code Execution Successful ===
```

2. Create a lambda to find the square of a number.

```
# lambda function to find square
square = lambda x: x ** 2

# take user input
num = float(input("Enter a number: "))

print("Square of the number:", square(num))
```

```
Enter a number: 4
Square of the number: 16.0

=== Code Execution Successful ===
```

3. Write a lambda that checks whether a number is even or odd.

```
# lambda function to check even or odd
even_odd = lambda x: "Even" if x % 2 == 0 else "Odd"

# take user input
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

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

print(f"The number {num} is {even_odd(num)}.")
```

```
Enter a number: 5
The number 5 is Odd.

=== Code Execution Successful ===
```

4. Use map() with lambda to double all elements in a list.

```
nums_input = input("Enter numbers separated by spaces: ")
numbers = []

for n in nums_input.split():
    numbers.append(int(n))  # convert each input to integer

# use map() with lambda to double elements
doubled = list(map(lambda x: x * 2, numbers))

print("Original list:", numbers)
print("List after doubling each element:", doubled)
```

```
Enter numbers separated by spaces: 3 4 5 6 7
Original list: [3, 4, 5, 6, 7]
List after doubling each element: [6, 8, 10, 12, 14]

=== Code Execution Successful ===
```

5. Use filter() with lambda to extract only even numbers from a list.

```
# take user input
nums_input = input("Enter numbers separated by spaces: ")
numbers = []

for n in nums_input.split():
    numbers.append(int(n))  # convert each input to integer

# use filter() with lambda to get even numbers
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))

print("Original list:", numbers)
print("Even numbers:", even_numbers)
```

```
Enter numbers separated by spaces: 2 4 6 8 9
Original list: [2, 4, 6, 8, 9]
Even numbers: [2, 4, 6, 8]
```

6.Use reduce() (from functools) with lambda to find the product of all numbers in a list.

```
from functools import reduce

# take user input
nums_input = input("Enter numbers separated by spaces: ")
numbers = []

for n in nums_input.split():
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

```
    numbers.append(int(n))  # convert each input to integer

# use reduce() with lambda to find product
product = reduce(lambda x, y: x * y, numbers)

print("Numbers in the list:", numbers)
print("Product of all numbers:", product)
```

```
Enter numbers separated by spaces: 1 2 3 4
Numbers in the list: [1, 2, 3, 4]
Product of all numbers: 24

=== Code Execution Successful ===
```

7. Create a lambda that returns the maximum of two numbers.

```
# lambda function to find maximum of two numbers
maximum = lambda a, b: a if a > b else b

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

print("The maximum number is:", maximum(a, b))
```

```
Enter first number: 6
Enter second number: 8
The maximum number is: 8.0

=== Code Execution Successful ===
```

8. Write a lambda to reverse a string.

```
# lambda function to reverse a string
reverse_string = lambda s: s[::-1]

# take user input
text = input("Enter a string: ")

print("Reversed string:", reverse_string(text))
```

```
Enter a string: python
Reversed string: nohtyp

=== Code Execution Successful ===
```

9. Use a lambda to sort a list of tuples based on the second element.

```
# take user input
n = int(input("Enter the number of tuples: "))
tuples_list = []

for i in range(n):
    t = input(f"Enter tuple {i+1} (two values separated by space): ").split()
    # converting tuple elements to int if possible
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

```
    try:
        t = (int(t[0]), int(t[1]))
    except ValueError:
        t = (t[0], t[1])
    tuples_list.append(t)

# sort the list of tuples based on the second element
sorted_list = sorted(tuples_list, key=lambda x: x[1])

print("\nOriginal list of tuples:", tuples_list)
print("Sorted list based on second element:", sorted_list)
```

```
Enter the number of tuples: 3
Enter tuple 1 (two values separated by space): 3 4
Enter tuple 2 (two values separated by space): 2 5
Enter tuple 3 (two values separated by space): 6 8

Original list of tuples: [(3, 4), (2, 5), (6, 8)]
Sorted list based on second element: [(3, 4), (2, 5), (6, 8)]

=== Code Execution Successful ===
```

10. Write a lambda function that converts all strings in a list to uppercase.

```
# take user input
words = input("Enter words separated by spaces: ").split()

# lambda function with map() to convert all strings to uppercase
uppercase_words = list(map(lambda s: s.upper(), words))

print("Original list:", words)
print("List in uppercase:", uppercase_words)
```

```
Enter words separated by spaces: app bake cake
Original list: ['app', 'bake', 'cake']
List in uppercase: ['APP', 'BAKE', 'CAKE']

=== Code Execution Successful ===
```

11. Create a lambda to find the cube of a number.

```
# lambda function to find cube
cube = lambda x: x ** 3

# take user input
num = float(input("Enter a number: "))

print("Cube of the number:", cube(num))
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

```
Enter a number: 4
Cube of the number: 64.0

=== Code Execution Successful ===
```

12. Write a lambda to check whether a string starts with a vowel.

```
# lambda function to check if a string starts with a vowel
starts_with_vowel = lambda s: s[0].lower() in "aeiou"

# take user input
text = input("Enter a string: ")

if starts_with_vowel(text):
    print("The string starts with a vowel.")
else:
    print("The string does not start with a vowel.")
```

```
Enter a string: Rural
The string does not start with a vowel.

=== Code Execution Successful ===
```

13.Use map() and lambda to convert a list of temperatures from Celsius to Fahrenheit.

```
# take user input
celsius_input = input("Enter temperatures in Celsius (separated by spaces): ")
celsius_list = []

for c in celsius_input.split():
    celsius_list.append(float(c))  # convert each input to float

# use map() with lambda to convert Celsius → Fahrenheit
fahrenheit_list = list(map(lambda c: (c * 9/5) + 32, celsius_list))

print("Temperatures in Celsius:", celsius_list)
print("Temperatures in Fahrenheit:", fahrenheit_list)
```

```
Enter temperatures in Celsius (separated by spaces): 40
Temperatures in Celsius: [40.0]
Temperatures in Fahrenheit: [104.0]

=== Code Execution Successful ===
```

14. Use filter() and lambda to find names longer than 5 characters in a list.

```
# take user input
names = input("Enter names separated by spaces: ").split()

# use filter() with lambda to find names longer than 5 characters
long_names = list(filter(lambda name: len(name) > 5, names))

print("Original list of names:", names)
print("Names longer than 5 characters:", long_names)
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

```
Enter names separated by spaces: blacks shakes
Original list of names: ['blacks', 'shakes']
Names longer than 5 characters: ['blacks', 'shakes']

=== Code Execution Successful ===
```

15. Use a lambda and map() to calculate the square root of numbers in a list.

```
# take user input
nums_input = input("Enter numbers separated by spaces: ")
numbers = []

for n in nums_input.split():
    numbers.append(float(n))  # convert each input to float

# use map() with lambda to calculate square roots (no math module)
square_roots = list(map(lambda x: x ** 0.5, numbers))

print("Original list:", numbers)
print("Square roots:", square_roots)
```

```
Enter numbers separated by spaces: 64
Original list: [64.0]
Square roots: [8.0]

=== Code Execution Successful ===
```

#Recursion

1.Write a recursive function to calculate the factorial of a number.

```
def factorial(n):
    if n == 0 or n == 1:   # base case
        return 1
    else:
        return n * factorial(n - 1)   # recursive call

# take user input
num = int(input("Enter a number: "))
print("Factorial of", num, "is:", factorial(num))
```

```
Enter a number: 4
Factorial of 4 is: 24

=== Code Execution Successful ===
```

2. Write a recursive function to generate the Fibonacci series up to n terms.

```
def fibonacci(n):
    if n <= 1:        # base case
        return n
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)   # recursive relation

# take user input
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

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

print("Fibonacci series up to", num, "terms:")

for i in range(num):
    print(fibonacci(i), end=" ")
```

```
Enter the number of terms: 8
Fibonacci series up to 8 terms:
0 1 1 2 3 5 8 13
=== Code Execution Successful ===
```

3. Create a recursive function to find the sum of natural numbers up to n.

```
def sum_natural(n):
    if n == 0:              # base case
        return 0
    else:
        return n + sum_natural(n - 1)   # recursive case

# take user input
num = int(input("Enter a number: "))
print("Sum of natural numbers up to", num, "is:", sum_natural(num))
```

```
Enter a number: 5
Sum of natural numbers up to 5 is: 15

=== Code Execution Successful ===
```

4. Write a recursive function to reverse a string.

```
def reverse_string(s):
    if len(s) == 0:             # base case
        return s
    else:
        return reverse_string(s[1:]) + s[0]   # recursive case

# take user input
text = input("Enter a string: ")
print("Reversed string:", reverse_string(text))
```

```
Enter a string: noon
Reversed string: noon

=== Code Execution Successful ===
```

5. Write a recursive function to count the digits in a number.

```
def count_digits(n):
    if n == 0:
        return 0
    else:
        return 1 + count_digits(n // 10)   # remove the last digit and count

# take user input
num = int(input("Enter a number: "))
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

```
# handle the case when input is 0
if num == 0:
    print("Number of digits: 1")
else:
    print("Number of digits:", count_digits(num))
```

```
Enter a number: 12345
Number of digits: 5

=== Code Execution Successful ===
```

6. Create a recursive function to calculate the power of a number (x^y).

```
def power(x, y):
    if y == 0:            # base case
        return 1
    else:
        return x * power(x, y - 1)   # recursive step

# take user input
base = float(input("Enter the base number (x): "))
exponent = int(input("Enter the exponent (y): "))

print(f"{base} raised to the power {exponent} is:", power(base, exponent))
```

```
Enter the base number (x): 2
Enter the exponent (y): 3
2.0 raised to the power 3 is: 8.0

=== Code Execution Successful ===
```

7. Write a recursive function to find the greatest common divisor (GCD) of two numbers.

```
def gcd(a, b):
    if b == 0:              # base case
        return a
    else:
        return gcd(b, a % b)   # recursive case (Euclidean algorithm)

# take user input
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))

print(f"GCD of {num1} and {num2} is:", gcd(num1, num2))
```

```
Enter first number: 54
Enter second number: 24
GCD of 54 and 24 is: 6

=== Code Execution Successful ===
```

8. Create a recursive function to check whether a number is palindrome or not.

```
def reverse_number(n, rev=0):
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

```
    if n == 0:                  # base case
        return rev
    else:
        return reverse_number(n // 10, rev * 10 + n % 10)   # recursive step

def is_palindrome(n):
    return n == reverse_number(n)

# take user input
num = int(input("Enter a number: "))

if is_palindrome(num):
    print(num, "is a palindrome.")
else:
    print(num, "is not a palindrome.")
```

```
Enter a number: 12345
12345 is not a palindrome.

=== Code Execution Successful ===
```

9. Write a recursive function to find the sum of elements in a list.

```
def sum_of_list(lst):
    if len(lst) == 0:              # base case
        return 0
    else:
        return lst[0] + sum_of_list(lst[1:])   # recursive case

# take user input
nums_input = input("Enter numbers separated by spaces: ")
numbers = []

for n in nums_input.split():
    numbers.append(float(n))  # convert each input to float

print("Sum of elements in the list:", sum_of_list(numbers))
```

```
Enter numbers separated by spaces: 5 10
Sum of elements in the list: 15.0

=== Code Execution Successful ===
```

10. Write a recursive function to print all elements of a list in reverse order.

```
def print_reverse(lst, index=None):
    if index is None:
        index = len(lst) - 1   # start from last index

    if index < 0:              # base case
        return
    else:
        print(lst[index], end=" ")
        print_reverse(lst, index - 1)   # recursive call with previous index

# take user input
nums_input = input("Enter elements separated by spaces: ")
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

```
elements = nums_input.split()

print("List elements in reverse order:")
print_reverse(elements)
```

```
Enter elements separated by spaces: A B C D E
List elements in reverse order:
E D C B A
=== Code Execution Successful ===
```

11. Create a recursive function to find the length of a string.

```
def string_length(s):
    if s == "":              # base case: empty string
        return 0
    else:
        return 1 + string_length(s[1:])   # count 1 and call for remaining string

# take user input
text = input("Enter a string: ")
print("Length of the string:", string_length(text))
```

```
Enter a string: abcde
Length of the string: 5

=== Code Execution Successful ===
```

12. Write a recursive function to convert a decimal number to binary.

```
def decimal_to_binary(n):
    if n == 0:                  # base case
        return ""
    else:
        return decimal_to_binary(n // 2) + str(n % 2)   # recursive step

# take user input
num = int(input("Enter a decimal number: "))

# handle the case when number is 0
if num == 0:
    print("Binary: 0")
else:
    print("Binary:", decimal_to_binary(num))
```

```
Enter a decimal number: 3
Binary: 11

=== Code Execution Successful ===
```

13. Create a recursive function to find the minimum element in a list.

```
def find_min(lst):
    if len(lst) == 1:               # base case: only one element left
        return lst[0]
    else:
        min_rest = find_min(lst[1:])   # recursive call on the rest of the list
```

Name: Sonali Shintre
Roll No: C-29
Cource No:SPP II(Python)

```
        return lst[0] if lst[0] < min_rest else min_rest

# take user input
nums_input = input("Enter numbers separated by spaces: ")
numbers = []

for n in nums_input.split():
    numbers.append(float(n))  # convert input to float

print("Minimum element in the list:", find_min(numbers))
```

```
Enter numbers separated by spaces: 3 4 5 6
Minimum element in the list: 3.0

=== Code Execution Successful ===
```

14. Write a recursive function to compute the sum of digits of a number.

```
def sum_of_digits(n):
    if n == 0:                # base case
        return 0
    else:
        return (n % 10) + sum_of_digits(n // 10)   # recursive case

# take user input
num = int(input("Enter a number: "))
print("Sum of digits:", sum_of_digits(num))
```

```
Enter a number: 12
Sum of digits: 3

=== Code Execution Successful ===
```

15. Create a recursive function to check whether a given string is palindrome.

```
def is_palindrome(s):
    if len(s) <= 1:             # base case
        return True
    elif s[0].lower() != s[-1].lower():   # compare first and last characters
        return False
    else:
        return is_palindrome(s[1:-1])     # recursive call for the middle substring

# take user input
text = input("Enter a string: ")

if is_palindrome(text):
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

```
Enter a string: madam
The string is a palindrome.

=== Code Execution Successful ===
```