Shreya Nair-24BIT196

AIM: To understand and apply functions and recursion in Python for modular, reusable, and efficient code writing.

HARDWARE & SOFTWARE REQUIREMENTS: Hardware:16GB RAM, Intel Processor(i9), Software: Python (Version 3.x), Google Colab (Cloud-based)

SYSTEM CONFIGURATION: Operating System: Windows 11, IDE: Google Colab

THEORY: A function is a block of reusable code designed to perform a specific task.It improves readability, structure, and reusability of code. A recursive function is one that calls itself to solve smaller instances of the problem. Functions are essential for breaking large programs into smaller, manageable parts and recursion is powerful for problems like factorial, Fibonacci, backtracking, etc.

REFERENCES:Geeks for Geeks, Python Documentation: https://docs.python.org/3/

1)Write a program that defines a function count_lower_upper() that accepts a string and calculates the number of uppercase and lowercase alphabets in it. It should return these values as a dictionary. Call this function for some sample string.

```
def count_lower_upper(s):
    result = {"UPPER": 0, "LOWER": 0}
    for char in s:
        if char.isupper():
            result["UPPER"] += 1
        elif char.islower():
            result["LOWER"] += 1
    return result

print(count_lower_upper("HelloPython"))
```

```
{'UPPER': 2, 'LOWER': 9}
```

2) Write a program that defines a function compute() that calculates the value of n + nn + nnn + nnnn, where n is digit received by the function. test the function for digits 4 to 7.

```
def compute(n):
    n = str(n)
    return int(n) + int(n*2) + int(n*3) + int(n*4)

for i in range(4, 8):
    print(f"{i} + {i*2} + {i*3} + {i*4} = {compute(i)}")
```

```
4 + 8 + 12 + 16 = 4936
5 + 10 + 15 + 20 = 6170
6 + 12 + 18 + 24 = 7404
7 + 14 + 21 + 28 = 8638
```

3) Write a program that defines a function create_array() to create and return a 3D array whose dimensions are passed to the function. Also initialize each element of this aray to a value passed to the function. e.g. create_array(3,4,5,n) where first three arguments are 3D array dimensions and 4th value is for initialing each value of the 3D array

```
def create_array(x, y, z, val):
    return [[[val for _ in range(z)] for _ in range(y)] for _ in range(x)]

arr = create_array(3, 4, 5, 1)
print(arr)
```

```
[[[1, 1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1, 1]], [[1, 1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1,
```

4) Write a program that defines a function sum_avg() to accept marks of five subjects and calculates total and average. It should return directly both values.

```
def sum_avg(marks):
    total = sum(marks)
    average = total / len(marks)
    return total, average

marks = [90, 80, 99, 100, 88]
print("Total and Average:", sum_avg(marks))
```

```
Total and Average: (457, 91.4)
```

5) Pangram is a sentence that uses every letter of the alphabet. Write a program to check whether a given string is pangram or not, through a user-defined function ispangram(). Test the function with “The quick brown fox jumps over the lazy dog” or “Crazy Fredrick bought many very exquisite opal jewels”. Hint: use set() to convert the string into a set of characters present in the string and use <= to check whether alphaset is a subset of the given string.

```
def ispangram(s):
    alphabet = set("abcdefghijklmnopqrstuvwxyz")
    return alphabet <= set(s.lower())

print(ispangram("The quick brown fox jumps over the lazy dog"))
```

```
True
```

6) Write a function to create and return a list containing tuples of the form (x,x2,x3) for all x between 1 and given ending value (both inclusive).

```
def power_tuples(n):
    return [(x, x**2, x**3) for x in range(1, n+1)]
print(power_tuples(9))
```

```
[(1, 1, 1), (2, 4, 8), (3, 9, 27), (4, 16, 64), (5, 25, 125), (6, 36, 216), (7, 49, 343), (8, 64, 512), (9, 81, 729)]
```

7) A palindrome is a word or phrase that reads the same in both directions. Write a program that defines a function ispalindrome() which checks whether a given string is a palindrome or not. Ignore spaces and case mismatch while checking for palindrome.

```
def ispalindrome(s):
    s = ''.join(s.lower().split())
    return s == s[::-1]

print(ispalindrome("Madam"))
```

```
True
```

8) Write a program that defines a function convert() that receives a string containing a sequence of whitespace separated words and returns a string after removing all duplicate words and sorting them alphanumerically. Hint: use set(), list () , sorted(), join().

```
def convert(s):
    words = set(s.split())
    return ' '.join(sorted(words))

print(convert("hello world hello python world"))
```

```
hello python world
```

9) Write a program that defines a function count_alpha_digits() that accepts a string and calculates the number of alphabets and digits in it. It should return these values as a dictionary.

```
def count_alpha_digits(s):
    result = {"ALPHA": 0, "DIGITS": 0}
    for char in s:
        if char.isalpha():
            result["ALPHA"] += 1
        elif char.isdigit():
            result["DIGITS"] += 1
    return result
```

```
print(count_alpha_digits("world98765fun7854"))
```

```
{'ALPHA': 8, 'DIGITS': 9}
```

10) Write a program that defines a function called frequency() which computes the frequency of words present in a string passed to it. The frequencies should be returned in sorted order of words in the string.

```
def frequency(s):
    words = s.split()
    freq = {}
    for word in words:
        freq[word] = freq.get(word, 0) + 1
    return dict(sorted(freq.items()))

print(frequency("this is a test this is only a test"))
```

```
{'a': 2, 'is': 2, 'only': 1, 'test': 2, 'this': 2}
```

11) Write a function create_list() that creates and returns a list which is an intersection of two lists passed to it.

```
def create_list(list1, list2):
    return [x for x in list1 if x in list2]

print(create_list([1, 2, 3, 4], [3, 4, 5, 6]))
```

```
[3, 4]
```

12) If a positive integer is entered through the keyword, write a recursive function to obtain the prime factors of the number

```
def prime_factors(n, f=2):
    if f > n:
        return []
    if n % f == 0:
        return [f] + prime_factors(n // f, f)
    return prime_factors(n, f + 1)

print(prime_factors(150))
```

```
[2, 3, 5, 5]
```

13) A positive integer is entered through the keyboard. Write a function to find its binary equivalent of this number.

```
def to_binary(n):
    if n == 0:
        return ""
    return to_binary(n // 2) + str(n % 2)

print(to_binary(11))
```

```
1011
```

14) A string is entered through the keyboard. Write a recursive function that counts the number of vowels in this string.

```
def count_vowels(s):
    if not s:
        return 0
    return (1 if s[0].lower() in 'aeiou' else 0) + count_vowels(s[1:])

print(count_vowels("hardwork"))
```

```
2
```

15) Write a recursive function that reverses the list of numbers that it receives.

```
def reverse_list(lst):
    if len(lst) <= 1:
        return lst
    return [lst[-1]] + reverse_list(lst[:-1])

print(reverse_list([1, 2, 3, 4, 5]))
```

```
[5, 4, 3, 2, 1]
```

16) Calculate ab where a and b received through the keyword using recursion.

```
def power(a, b):
    if b == 0:
        return 1
    return a * power(a, b - 1)

print(power(6, 5))
```

```
7776
```

17) A list contains some negative and some positive values. Write a recursive function that sanitizes the list by replacing all negative numbers with 0.

```
def sanitize(lst):
    if not lst:
        return []
    return [0 if lst[0] < 0 else lst[0]] + sanitize(lst[1:])

print(sanitize([1, -2, 3, -4, 5]))
```

```
[1, 0, 3, 0, 5]
```

18) Write a recursive function to obtain average of all numbers present in a given list.

```
def avg(lst):
    def helper(lst, total, count):
        if not lst:
            return total / count
        return helper(lst[1:], total + lst[0], count + 1)
    return helper(lst, 0, 0)

print(avg([15, 20, 35, 40]))
```

```
27.5
```

19) Write a recursive function to obtain length of a given string.

```
def length(s):
    if s == "":
        return 0
    return 1 + length(s[1:])

print(length("Palindrome"))
```

```
10
```