**Functions and recursions:**

**Aim**

To solve problems using function and recursions

**Q.No 1 :** 1. Define a user-defined function for finding the sum return the sum of the digits. Get input from the user and pass the arguments while calling the function

Python code:

#To find the sum of digits of a number using recursion:

num=int(input("Enter the number"))

def lenofnum(num):

rem=num%10

if num==0:

return 0

else:

return (rem+ lenofnum(num//10))

print(lenofnum(num))

Test cases:

Enter the number45678

30

Enter the number76

13

QNo. 2. Define user-define functions separate and prime ,which get the input from the user and separate the digits, then print the digits that are prime.( Usage of keyword is mandatory).

Python code:

#program to print prime digits

def separation(n):

while n:

last = n%10

n//=10

prime = prime\_finder(last)

if prime:

print(last)

def prime\_finder(digit):

prime = True

for i in range(2, (digit//2)+1):

if digit%i == 0:

prime = False

break

return prime

n = int(input("Enter a number: "))

separation(n)

Test cases:

Enter a number: 678976

7

7

3. Generate the range of “N” values and find a. the sum of odd values b. the sum of even values

Python code:

#program to find sum of odd and even in a range

def find\_sum\_odd(n):

\_sum = 0

for i in range(n+1):

if i%2 == 1:

\_sum += i

return \_sum

def find\_sum\_even(n):

\_sum = 0

for i in range(n+1):

if i%2 == 0:

\_sum += i

return \_sum

n = int(input("Enter the value of N: "))

odd = find\_sum\_odd(n)

even = find\_sum\_even(n)

print("The sum of odd numbers in the range is", odd)

print("The sum of even numbers in the range is", even)

Test cases:

Enter the value of N: 76543

The sum of odd numbers in the range is 1464745984

The sum of even numbers in the range is 1464707712

========================== RESTART: D:/sumofoddeve.py ==========================

Enter the value of N: 10

The sum of odd numbers in the range is 25

The sum of even numbers in the range is 30

4. Pass a number to a function as parameter which finds the factors of the given number and displays it. If a number is not given, then display with default value.

Python code:

# program to finds the factors of the given number and displays it

def factors(n=10):

for i in range(1, (n//2)+1):

if n%i == 0:

print(i, "is a factor of", n)

choice = input("Want to give a number? (yes/no): ")

if choice == "yes":

n = int(input("Enter a number: "))

factors(n)

else:

factors()

Test cases:

Want to give a number? (yes/no): yes

Enter a number: 56

1 is a factor of 56

2 is a factor of 56

4 is a factor of 56

7 is a factor of 56

8 is a factor of 56

14 is a factor of 56

28 is a factor of 56

Want to give a number? (yes/no): yes

Enter a number: 68

1 is a factor of 68

2 is a factor of 68

4 is a factor of 68

17 is a factor of 68

34 is a factor of 68

5. Define user-defined function which computes the sum of the sine series. Recursive:

Python code:

#program to find sine series

def fact(i):

if i == 0:

return 1

return i \* fact(i-1)

def sine\_series(x, c, n, i):

if i == n:

return 0

else:

if i % 2 == 0:

return -(x \*\* c) / fact(c) + sine\_series(x, c+2, n, i+1)

else:

return (x \*\* c) / fact(c) + sine\_series(x, c+2, n, i+1)

x = float(input("Enter the angle in radians: "))

n = int(input("Enter the number of terms to appoximate: "))

print("The approx sine value is: ", sine\_series(x, 1, n, 1))

Test cases:

Enter the angle in radians: 1

Enter the number of terms to appoximate: 6

The approx sine value is: 0.8414710097001764

Enter the angle in radians: 4

Enter the number of terms to appoximate: 6

The approx sine value is: -0.6617283950617274

6. Write a Python program to calculate factorial of a given number using recursive function. The base case should handle the negative integers by printing an error message and returns none to indicate that something went wrong.

Python code:

#program to find factorial recursively

def fact(n):

if n < 0:

print("Can't find factorial for negatives")

return -1

if n <= 1:

return 1

return n \* fact(n-1)

n = int(input("Enter the number: "))

res = fact(n)

if res != -1:

print("The factorial of the given number is ", res)

Test cases:

Enter the number: 7

The factorial of the given number is 5040

======================= RESTART: D:/factorialrec.py =======================

Enter the number: 3

The factorial of the given number is 6

7. Write a Python program to solve the Fibonacci sequence using recursion

Python code:

# program to solve Fibonacci sequence using recursion

def fib(n):

if n <= 1:

return n

else:

return(fib(n-1) + fib(n-2))

n = int(input("Enter a number: "))

res = fib(n)

print("The value of", n, "th digit in Fibonacci sequence is", res)

Test cases:

Enter a number: 8

The value of 8 th digit in Fibonacci sequence is 21

. 8. Write a Python program to get the sum of a non-negative integer. Test Data: sumDigits(345) -> 12, sumDigits(45) -> 9

Python code:

#To find the sum of digits of a number using recursion:

num=int(input("Enter the number"))

def lenofnum(num):

rem=num%10

if num==0:

return 0

else:

return (rem+ lenofnum(num//10))

print(lenofnum(num))

Test cases:

Enter the number45678

30

Enter the number76

13

9. Write a Python program to calculate the sum of the positive integers of n+(n-2)+(n-4)... (until n-x =< 0). Test Data: sum\_series(6) -> 12, sum\_series(10) -> 30

Python code:

#program to calculate the sum of the positive integers of n+(n-2)+(n-4)... (until n-x =< 0).

def \_sum(n, x=0):

if n-(2\*x) <= 0: #base case where the element become 0 or negativeS

return 0

return n-(2\*x) + \_sum(n, x+1)

n = int(input("Enter the value of n: "))

res = \_sum(n)

print("The sum of the series is", res)

Testcases:

Enter the value of n: 5

The sum of the series is 9

10. The greatest common divisor (GCD) of a and b is the largest number that divides both of them with no remainder. One way to find the GCD of two numbers is based on the observation that if r is the remainder when a is divided by b, then gcd(a, b) = gcd(b, r). As a base case, we can use gcd(a, 0) = a. Write a recursive function called gcd that takes parameters a and b and returns their greatest common divisor. Note : In all the above cases for recursive functions, 1. Identify the base condition (or) termination condition 2. Recursively call the function within the body of function such that it reaches the base condition to terminate itself. 3. In the function definition, include comments to mention the assumptions about the input parameter. 4. For the input other than from the expected domain, handle with appropriate code

Python code:

# program to find gcd using recursion

def gcd(a, b): #assumed where a is greater than b

if b == 0:

return a

return gcd(b, a%b)# eulers method

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

b = int(input("Enter another number: "))

if a > b:

res = gcd(a, b)

else:

res = gcd(b, a)

print("The gcd of given number is", res)

Test cases:

Enter a number: 76

Enter another number: 54

The gcd of given number is 2

========================================= RESTART: D:/gcd.py ========================================

Enter a number: 73

Enter another number: 32

The gcd of given number is 1

. Part – B (Optional) Write a function “perfect()” that determines if the parameter number is a perfect number. Use this function in a program that determines and prints all the perfect numbers between 1 and 1000. [An integer number is said to be a “perfect number” if its factors, including 1(but not the number itself), sum to the number. E.g., 6 is a perfect number because 6=1+2+3].

Python code:

#program that determines and prints all the perfect numbers between 1 and 1000.

def perfect(n):

\_sum = 0

for i in range(1, (n//2)+1):

if n%i == 0:

\_sum += i

if \_sum == n:

print(n)

for i in range(1, 1001):

perfect(i)

Test cases:

6

28

496

2. Write python code for reversing a string, then replace the occurrence of vowels with “$”. Note: Don’t use Slicing methods and built-in functions for reversing and replacing.

Python Program:

#program for reversing a string, then replace the occurrence of vowels with “$”.

def reverse(string):

rev = ''

for i in range(len(string)-1, -1, -1):#accessing the string from reverse

rev += string[i]

print("The string after reversing:", rev)

replace(rev)

def replace(string):

res=''

for i in range(len(string)):

if string[i] in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']: #chekcing whether current character is vowel

res+='$'

else:

res+=string[i]

print("The string after repalcement:", res)

string = input("Enter a string: ")

reverse(string)

Test cases:

Enter a string: Tokillamockingbird

The string after reversing: dribgnikcomallikoT

The string after repalcement: dr$bgn$kc$m$ll$k$T

3. Check whether a given number is a happy or sad number. (Hint: A number (positive integer) is called a happy number when it is replaced by the sum of the squares of its digits on a repeated basis until the sum of the squares of its digits equals 1.) Ex: Happy Number 23 is a happy number. Let’s see how. 23 = 22 + 32 = 4 + 9 = 13 (sum of the squares of its digits) 13 = 12 + 32 = 1 + 9 = 10 (sum of the squares of the digits) 10 = 12 + 02 = 1 (sum of the squares of the digits) Ex. Sad Number 36 is an unhappy number. Let’s see how. 36 = 32 + 62 = 9 + 36 = 45 45 = 42 + 52 = 16 + 25 = 41 41 = 42 + 12 = 16 + 1 = 17 17 = 12 + 72 = 1 + 49 =50 50 = 52 + 02 = 25 + 0 = 25 25 = 22 + 52 = 4 + 25 = 29 29 = 22 + 92 = 4 + 81 = 85 85 = 82 + 52 = 64 + 25 = 89 89 = 82 + 92 = 64 + 81 = 145 145 = 12 + 42 + 52 = 1 + 16 + 25 = 42 42 = 42 + 22 = 16 + 4 = 20 20 2 + 0 = 4 + 0 = 4

Python code:

#program to Check whether a given number is a happy or sad number.

def sqrDigit(n):

\_sum = 0

while n:

last = n%10

n//=10

\_sum += last\*last

return \_sum

def happyOrSad(n):

if n < 10:

if n == 1:

return True

return False

else:

happyOrSad(sqrDigit(n))

n = int(input("Enter a number: "))

happy = happyOrSad(n)

if happy:

print("Happy")

else:

print("Sad")

Test cases:

Enter a number: 676

Sad

4.The Ackermann function, A(m, n), is defined

As

Python code:

#program to find the value of Ackermann function

def ack(m, n):

if m == 0:

return n+1

elif m > 0 and n == 0:

return ack(m-1, 1)

elif m > 0 and n > 0:

return ack(m-1, ack(m, n-1))

m = int(input("Enter the value of m: "))

n = int(input("Enter the value of n: "))

print("The value of Ackermann function for", m, "and", n, "is:", ack(m,n))

Test cases:

Enter the value of m: 2

Enter the value of n: 3

The value of Ackermann function for 2 and 3 is: 9

**Learning Outcomes**

Due to the programs above the students learn functions and recursions in a great depth. Which increases the understanding of these topics