Harsh Kasliwal

2003085

C21

**EXPERIMENT 3**

**Q1. Write a Python function to check whether a number is perfect or not.**

(Note: The first perfect number is 6, because 1, 2, and 3 are its proper positive divisors, and 1 + 2 + 3 = 6. Equivalently, the number 6 is equal to half the sum of all its positive divisors: (1 + 2 + 3 + 6) / 2 = 6. The next perfect number is 28 = 1 +2 + 4 + 7 + 14. This is followed by the perfect numbers 496 and 8128.)

**Functions in Python:**

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into function. A function can return data as a result. In Python a function is defined using the def keyword In Python a function is defined using the def keyword:

def example (): #This defines it print("Example.") #This is the defined commands

Example ():

**Perfect Number:**

Perfect number is a number whose proper positive divisors add up to the number itself

**CODE:**

def perfect\_num(n):

sum=0

for i in range(1,n):

if(n % i ==0):

sum = sum + i

if(sum == n):

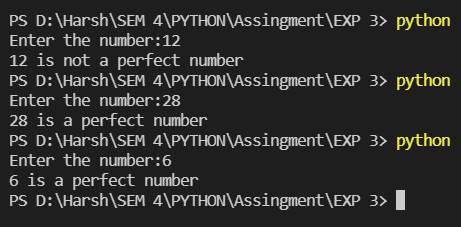
print("%d is a perfect number" %n)

else:

print("%d is not a perfect number" %n)

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

perfect\_num(num)



**Q2. Write a Python function to check whether a string is a pangram or not.**

(Note: Pangrams are words or sentences containing every letter of the alphabet at least once. For example: "The quick brown fox jumps over the lazy dog")

**Pangrams:**

Pangrams are words or sentences containing every letter of the alphabet at least once.

**• Lower() method:**

Converts every character to a lowercase character in a String.

**• set()method:**

Converts a list to a set in which duplicate elements are removed and only unique elements are present.

**• filter( function, iterable ) method:**

It filters the items if condition is true and removes if false. With the help of a function which returns boolean (True/False) and iterates on the iterable provided.

**• lambda function:**

A lambda function is a small anonymous function. A lambda function can take

any number of arguments, but can only have one expression.

Syntax:

lambda arguments: expression

Eg:

x = lambda a, b, c : a + b + c

print(x(5, 6, 2))

**CODE**

def pangram(str):

alphabet = "abcdefghijklmnopqrstuvwxyz"

for char in alphabet:

if char not in str.lower():

return 0

return 1

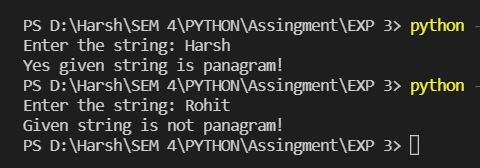
string = str(input("Enter the string: "))

if(pangram(string)==1):

print("Yes given string is panagram!")

else:

print("Given string is not panagram!")



**Q3. Python menu driven program to develop simple calculator using variable length argument**

**• \*variable\_name:**

If you do not know how many arguments that will be passed into your function, add a \* before the parameter name in the function definition.

**CODE:**

def add(\*num):

sum = 0

for num in nums:

sum =num + sum

return sum

def sub(\*num):

sum = 0

for num in nums:

sum=num-sum

return sum

def mul(\*num):

prod = 1

for num in nums:

prod =num \* prod

return prod

ch = int(input("1. Addition\n2. Subtraction\n3. Multiplication\n\nEnter your choice: "))

print("To stop entering numbers enter '!'")

nums = []

while True:

n=input()

if n =='!':

break

else:

nums.append(int(n))

if ch==1:

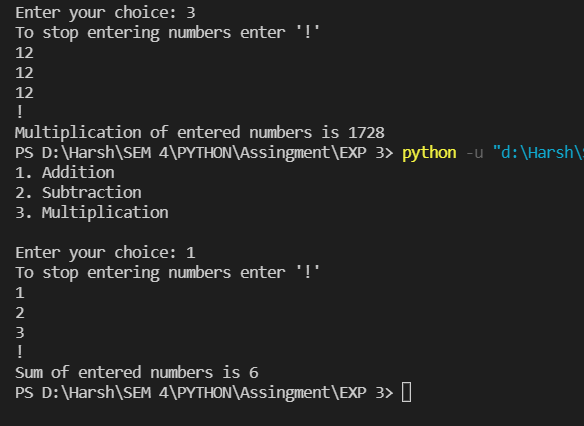
print("Sum of entered numbers is {}".format(add(\*nums)))

elif ch==2:

print("Subtraction of entered numbers is {}".format(sub(\*nums)))

elif ch==3:

print("Multiplication of entered numbers is {}".format(mul(\*nums)))



**Q4. Program to calculate factorial of a number using recursion.**

**CODE:**

def factorial(x):

if x==1:

return 1

else:

return(x\*factorial(x-1))

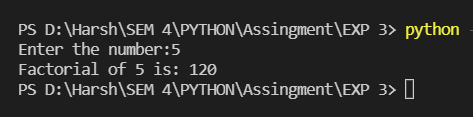
n=int(input("Enter the number:"))

if n<0:

print("Factorial of this number does not exist ")

else:

print("Factorial of {} is: {}".format(n,factorial(n)))



**Q5. Python program to calculate square and cube of a number and use two decorators, one to increase result by 4 and another to multiply result by 2.**

• **Decorator:**

A decorator takes in a function, adds some functionality and returns it. This is also called metaprogramming because a part of the program tries to modify another part of the program at compile time.

**CODE:**

def decor\_add(func):

def inner():

value1=func()

return value1+4

return inner

def decor\_multiply(func):

def inner():

value2=func()

return value2\*2

return inner

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

def square():

return n\*\*2

def cube():

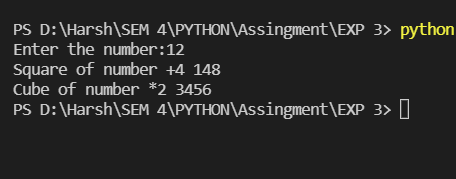
return n\*\*3

res1=decor\_add(square)

res2=decor\_multiply(cube)

print("Square of number +4 {}".format(res1()))

print("Cube of number \*2 {}".format(res2()))



**Q6. Write menu driven python program that accept list of numbers and performs following operation on list written in another module**

**• Summation of all elements**

**• Product of all elements**

**• Summation of elements at even indices**

**• add elements in the list**

**• Modules:**

Modules refer to a file containing Python statements and definitions. A file containing Python code, for example: example.py, is called a module, and its module name would be an example.

We use modules to break down large programs into small manageable and organized files.

**CODE:**

a=[]

def add\_all():

x=len(a)

sum=0

for i in range(0,x):

sum=sum+a[i]

print("Sum of all elements is {}" .format(sum))

def product\_all():

x=len(a)

prod=1

for i in range(0,x):

prod=prod\*a[i]

print("Product of all elements is {}" .format(prod))

def add\_at\_even():

x=len(a)

sum2=0

for i in range(0,x):

if i%2!=0:

sum2=sum2+a[i]

print("Sum of elements at even places is : {}".format(sum2))

def insert\_an\_element():

ele=int(input("Insert the element : "))

a.append(ele)

choice=0

while choice<5:

print("1.Add all elements\n2.Product of all elements\n3.Summation of elements at even indices\n4.Add elements in list\n5.exit")

choice=int(input("Select the option :"))

if choice==1:

add\_all()

elif choice==2:

product\_all()

elif choice==3:

add\_at\_even()

elif choice==4:

insert\_an\_element()

else:

print("Exitt")

