**LAB#10**

**(Function/ Tuples)**

**-**

Write a function ball\_collide that takes two balls as parameters and computes if they are

colliding. Your function should return a Boolean representing whether or not the balls are colliding. Hint: Represent a ball on a plane as a tuple of (x, y, r), r being the radius If (distance between two balls centers) <= (sum of their radii) then (they are colliding).

## Description:

We need to write a function to find out whether two balls are colliding or not by using the distance formulae i.e.,, If (distance between two balls centers) <= (sum of their radii) then we must return True. Otherwise, we must return False. Distance between two ball centers can be calculated by using the formulae:

Here, We are using the concept **functions**.

## Functions:

A function is a group of related statements that performs a specific task. Funtions helps break our program into smaller and modular chunks. As our program grows longer, functions make it more organized and manageable.

Further more, It avoids repetition and code reusable. The general form of a function is as follows:

def function\_name(parameter\_list): “””doc-string””” statement(s)

## Function Call:

Once we define a function, We can call it from another function, program or even from the Python prompt. To call a function, We can simply type the function name with appropriate paramenters.

## return statement:

This statement is used to exit a function and go back to place from where it was called. The general form of return statement is:

return [expression]



## Algorithm:

**Function ball\_collide((x1,y1,r1),(x2,y2,r2))**

**Input:** 2 tuples (containing co-ordinates and radii of two balls)

**Output**: Boolean Value Step1: Start

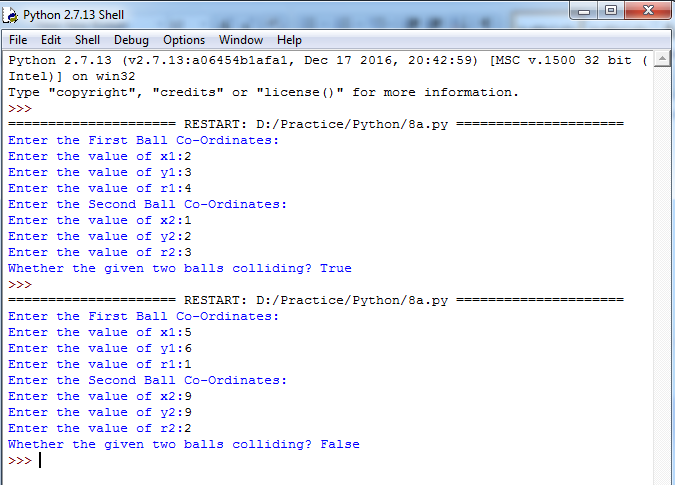
Step2: Calculate math.sqrt((x2-x1)\*\*2 + (y2-y1)\*\*2) and store the result in distance Step3: Add the radii r1 and r2 and store the result in sum\_radius

Step4: Check whether if distance is less than or equal to sum\_radius or not. If yes, goro step5.

Otherwise, goto Step6 Step5: return True and goto Step7 Step6: return False

Step7: Stop

## Output:





**Aim:**

**EXERCISE - 2**

Find mean, median, mode for the given set of numbers in a list.

## Description:

We need to write a function to find out mean, median and mode for a given set of numbers in a list. Mean, Median and Mode are 3 kinds of averages.

## Mean:

It is the average we are used to, where we add up all the numbers and then divide by the total number of numbers.

## Median:

It is the middle value in the list of numbers. To find median, all numbers have to be listed in numerical order from smallest to largest.

If total numbers are odd, then median is the middle value. If total numbers are even, then median is the average of 2 middle values.

## Mode:

It is the value that occurs most often. If no number is repeated among a list of numbers then there is no mode for the list.

## Algorithm:

**Function mean\_median\_mode(marks) Input:** A List

**Output**: Mean, Median and Mode Step1: Start

Step2: Initialize sum to 0

Step3: Calculate the length of the list marks and store the result in length

Step4: Calculate the addition of all numbers in the list and store the result in sum Step5: Divide sum and length and store the result in mean

Step6: Sort all elements in the list marks and store the result in the list marks1

Step7: Check whether the total number of elements in the list marks1 is odd or not. If yes, goto Step8. Otherwise, goto Step9

Step8: Assign middle element of the list marks1 to median and goto Step10



Step9: Calculate the average of middle two elements of the list marks1 and store the result in median

Step10: Create an empty dictionary dict\_mode and an empty list mode Step11: Repeat Steps 12 to 14 until end of the list marks is reached

Step12: Check whether the ith element of the list marks is present in the dictionary dict\_mode or not.

If yes, goto Step13. Otherwise, goto Step14

Step13: Increment value of the key element (ith element in the list marks) in the dictionary dict\_mode by 1 and continue with the next element in the list marks

Step14: Assign value of the key element (ith element in the list marks) in the dictionary dict\_mode to 1

Step15: Retrieve all values in the dictionary dict\_mode and store retrieved values in the list max\_frequency\_list

Step16: Find out maximum value in the list max\_frequency\_list and store the result value in max\_frequency

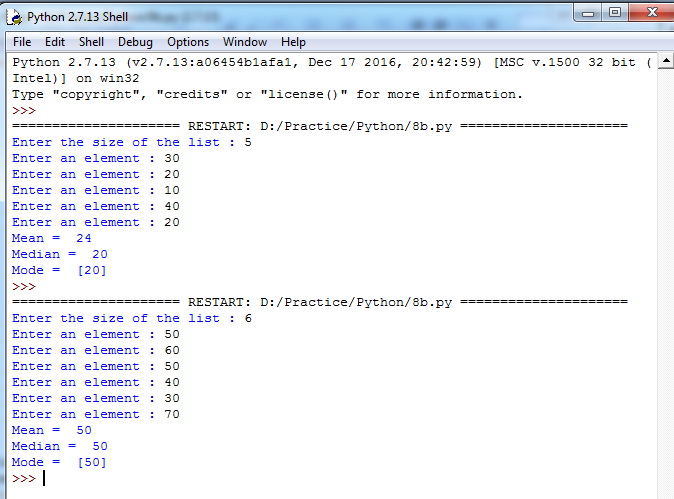
Step17: Repeat Steps 18 and 19 until end of the dictionary dict\_mode is reached

Step18: Compare value of ith key element with max\_frequency. If both are equal, then goto Step19 Step19: Add ith key element of the dictionary dict\_mode to the list mode

Step20: return the tuple which contains 3 values mean, median and mode Step21: Stop



## Output:



## Questions:

1. What is the importance of functions in python?
2. Why programmers need to choose a function programming in python?
3. What is function call and function definition?
4. What is parameter/argument?
5. What is pass by reference and pass by value in python?
6. What is the output of the below program? def printMax(a, b):

if a > b:

print(a, ' is maximum') elif a == b:

print(a, ' is equal to', b)

else:

printMax(3, 4)

print(b, ' is maximum')

1. How to retrieve all the values in a dictionary?
2. How to sort all the elements of a list in python?
3. What‟s the difference between lists and tuples?
4. What is the purpose of return statement?



## EXERCISE NO: 3

## (ITS IS OPTIONAL QUESTION / BONUS MARKS IF IT ATTEMPT)

**Aim:**

Write a function nearly\_equal to test whether two strings are nearly equal.

## Description:

Two strings are said to be nearly equal if a single mutation applied to one string will result in another string. That means, Given two strings s1 and s2, find if s1 can be converted to s2 with exactly one edit(mutation). If yes, then the function should return a True value as the result. Otherwise, it must return a False value as the result.

## Algorithm:

**Function nearly\_equal (s1, s2) Input:** Two Strings

**Output**: A Boolean value Step1: Start

Step2: Calculate the lengths of s1 ans s2 and store the results in m, n

Step3: Check whether the absolute difference between m and n is greater than 1 or not. If yes, goto Step4. Otherwise, goto Step5

Step4: return the boolean value False and goto Step22 Step5: Initialize count to i and j to 0

Step6: Repeat Steps 7 to 16 until i < m and j < n

Step7: Check whether ith character in s1 and jth character in s2 are equal or not. If yes, goto Step16.

Otherwise, goto Step8

Step8: Check whether the value of count is 1 or not. If yes, goto Step9. Otherwise, goto Step10 Step9: return the boolean value False and goto Step22

Step10: Check whether m is greater than n or not. If yes, goto Step11. Otherwise, goto Step12 Step11: Increment the value of i by 1 and goto Step15

Step12: Check whether m is less than n or not. If yes, goto Step13. Otherwise, goto Step14 Step13: Increment the value of j by 1 and goto Step15

Step14: Increment the values of i and j by 1

Step15: Increment the value of count by 1 and continue with the next iteration Step16: Increment the values of i and j by 1



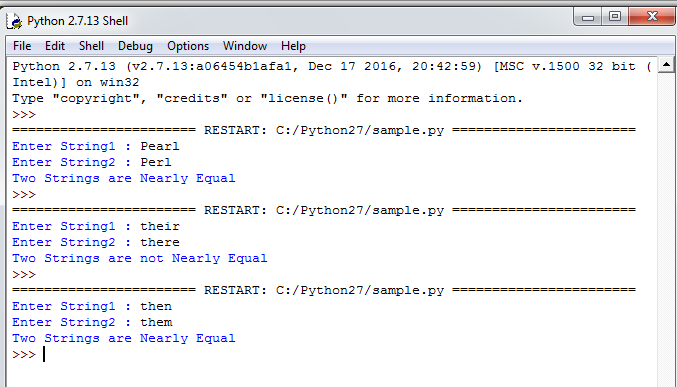
Step17: Check whether the condition (i < m or j < n) is True or not. If yes, goto Step18. Otherwise, goto Step19

Step18: Increment the value of count by 1

Step19: Check whether the value of count is 1 or not. If yes, goto Step20. Otherwise, goto Step21 Step20: return True and goto Step22

Step21: return False Step22: Stop

## Output:





**Aim:**

**EXERCISE - 4**

Write a function dups to find all duplicates in the list.

## Description:

We need to write a function dups to find all duplicate elements in the list. If an element is repeated more than once in the list, then add that repeated element to the resultant list.

## Algorithm:

**Function dups(elements) Input:** A List

**Output**: A List containing only duplicate elements Step1: Start

Step2: Create an empty dictionary dict\_dups and an empty list list\_dups Step3: Repeat Steps 4 to 6 until end of the list elements is reached

Step4: Check whether ith element in the list elements is present in the dictionary dict\_dups or not. If yes, goto Step5. Otherwise, goto Step6

Step5: Increment the value of key element (ith element of the list elements) in the dictionary dict\_dups by 1 and continue with next element of the list elements

Step6: Assign the value of key element (ith element of the list elements) in the dictionary dict\_dups to 1

Step7: Repeat Steps 8 and 9 until end of the dictionary dict\_dups is reached

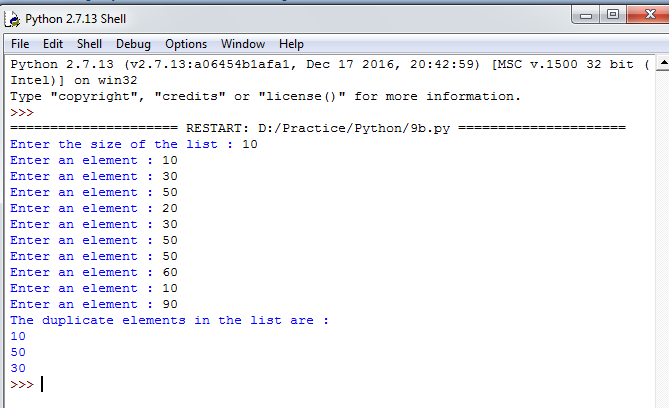
Step8: Check whether the value of ith key element in the dictionary dict\_dups is more than 1 or not.

If yes, goto Step9. Otherwise, Continue with the next element of the dictionary dict\_dups Step9: Add ith key element of the dictionary dict\_dups to the list list\_dups

Step10: return the list list\_dups Step11: Stop



## Output:





**Aim:**

**EXERCISE - 5**

Write a function unique to find all the unique elements of a list.

## Description:

We need to write a function unique to find all unique elements in the list. If an element is found only once in the list, then add that element to the resultant list.

## Algorithm:

**Function unique(elements) Input:** A List

**Output**: A List containing only unique elements Step1: Start

Step2: Create an empty dictionary dict\_ unique and an empty list list\_unique Step3: Repeat Steps 4 to 6 until end of the list elements is reached

Step4: Check whether ith element in the list elements is present in the dictionary dict\_unique or not.

If yes, goto Step5. Otherwise, goto Step6

Step5: Increment the value of key element (ith element of the list elements) in the dictionary dict\_unique by 1 and continue with next element of the list elements

Step6: Assign the value of key element (ith element of the list elements) in the dictionary dict\_unique to 1

Step7: Repeat Steps 8 and 9 until end of the dictionary dict\_unique is reached

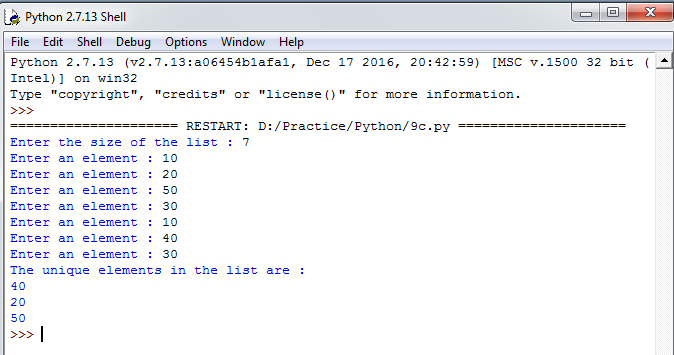
Step8: Check whether the value of ith key element in the dictionary dict\_unique is equal to 1 or not. If yes, goto Step9. Otherwise, Continue with the next element of the dictionary dict\_unique

Step9: Add ith key element of the dictionary dict\_unique to the list list\_unique Step10: return the list list\_unique

Step11: Stop



## Output:



## Questions:

1. What is recursive function?
2. What is scope?
3. What is default argument in python?
4. What is the output of the below program? x = 50

def func(x):

x = 2

func(x)

print 'x is now', x



1. What is the output of the below program? def power(x, y=2):

r = 1

for i in range(y):

r = r \* x

return r print power(3) print power(3, 3)

1. What is the output of the below program? def a(b):

b = b + [5]

c = [1, 2, 3, 4]

a(c) print(len(c))

1. What is the output of the below program? a=10

b=20

def change():

global b a=45 b=56

change() print(a) print(b)

1. What is the output of the below program? def change(one, \*two):

print(type(two)) change(1,2,3,4)

1. If a function doesn‟t have a return statement, which of the following does the function return?
2. What is the type of each element in sys.argv?

## Aim:

**EXERCISE - 7**

Write a function cumulative\_product to compute cumulative product of a list of numbers.

## Description:

We need to write a function cumulative\_product to find cumulative product of numbers in

the list.

A Cumulative product is a sequence of partial products of a given sequence. For example,

The cumulative product of sequence [a,b,c,…..] are a,ab,abc,…..

## Algorithm:

**Function cumulative\_product(numbers) Input:** A List

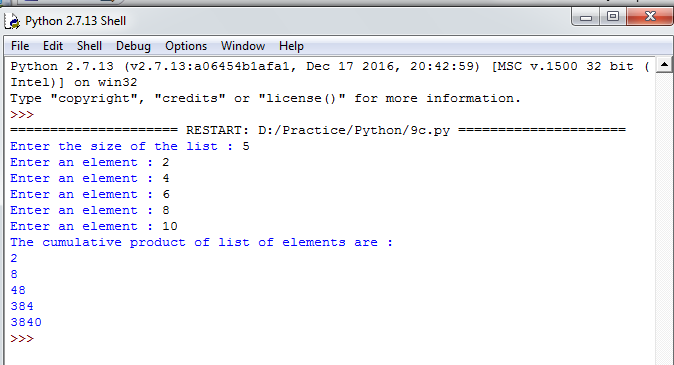
**Output**: Cumulative products of a numbers in a list Step1: Start

Step2: Create an empty list cum\_prod\_list and Initialize prod to 1 Step3: Repeat Steps 4 and 5 until end of the list numbers is reached Step4: Multiply prod with number and store the result in prod Step5: Add prod to the list cum\_prod\_list

Step6: return the list cum\_prod\_list Step7: Stop



## Output:





**Aim:**

**EXERCISE - 8**

Write a function reverse to reverse a list. Without using the reverse function.

## Description:

We need to write a function reverse to reverse the given elements in a list. Reversing of a list is done by using the feature called **slicing**.

Python‟s list objects have an interesting feature called slicing. We can view it as an extension of the square brackets indexing syntax. It includes a special case where slicing a list with “[::-1]” produces a reversed copy.

## Example:

My\_list = [10,20,30,40]

Rev\_list = My\_list[::-1]

Then, Rev\_list contains the list [40,30,20,10]

## Algorithm:

**Function reverse(elements) Input:** A List

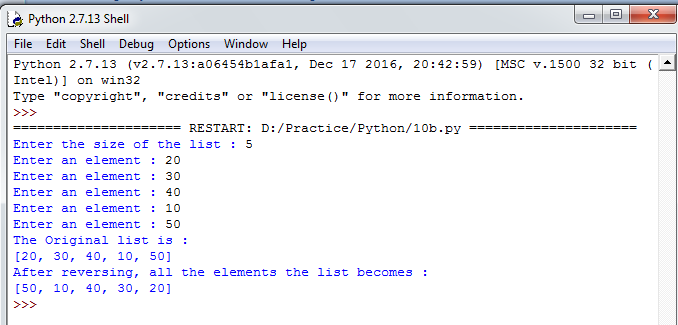
**Output**: A List (Obtained by Reversing the elements of an input list) Step1: Start

Step2: Reverse the given list by using slicing and store the result in the list reverse\_list Step3: return the list reverse\_list

Step4: Stop



## Output:





**Aim:**

line.

## EXERCISE - 9

Write function to compute gcd, lcm of two numbers. Each function shouldn‟t exceed one

## Description:

We need to write a functions to compute gcd, lcm of numbers. But here the constraint is each function should not exceed one line. For this, We are using **lambda function**.

## lambda function:

We can use lambda keyword to create small anonymous functions. lambda functions can have any number of arguments but contains only one expression. The expression is evaluated and returned. They cannot contains commands or multiple expressions.

The general form of lambda function is as followes:

## lambda arguments : expression

**Example:**

square = lambda x : x\*2

In the above statement, lambda x : x\*2 is the lambda function. Here, x is an argument and x\*2 is the expression that gets evaluated and returned.

## Algorithm:

**Function gcd(a,b) Input:** Two Numbers

**Output**: GCD of two numbers Step1: Start

Step2: Check whether the value of b is 0 or not. If yes, goto Step3. Otherwise, goto Step4 Step3: return the value of a and goto Step5

Step4: Call gcd(a,b) with a = b and b = a % b Step5: Stop

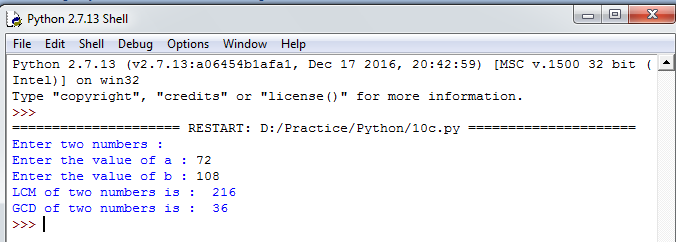
**Function lcm(a,b) Input:** Two Numbers

**Output**: LCM of two numbers Step1: Start



Step2: return (a \* b)//gcd(a,b) Step3: Stop

## Output:



## Questions:

1. What are lambda functions in Python?
2. What are fruitful functions in Python?
3. What is the data structure used to perform recursion?
4. What is tail recursion?
5. What actions are performed when a function is called?
6. What is the output of the code shown below? x = 5

def f1():

global x x = 4

def f2(a,b):

global x return a+b+x

f1()

total = f2(1,2) print(total)

1. What is the output of the code shown below? x=100

def f1():

global x x=90

def f2():

global x x=80

print(x)

1. Read the code shown below carefully and point out the global variables: y, z = 1, 2

def f():

global x x = y+z

1. What does f(120,13) return for the following function defintion? def f(m,n):

ans = 1

while (m - n >= 0):

(ans,m) = (ans\*2,m-n) return(ans)

1. What does g(9000,3) return for the following function definition? def g(x,y):

val = 0

while (x > y):

(val,x) = (val+1,x/y) return(val)