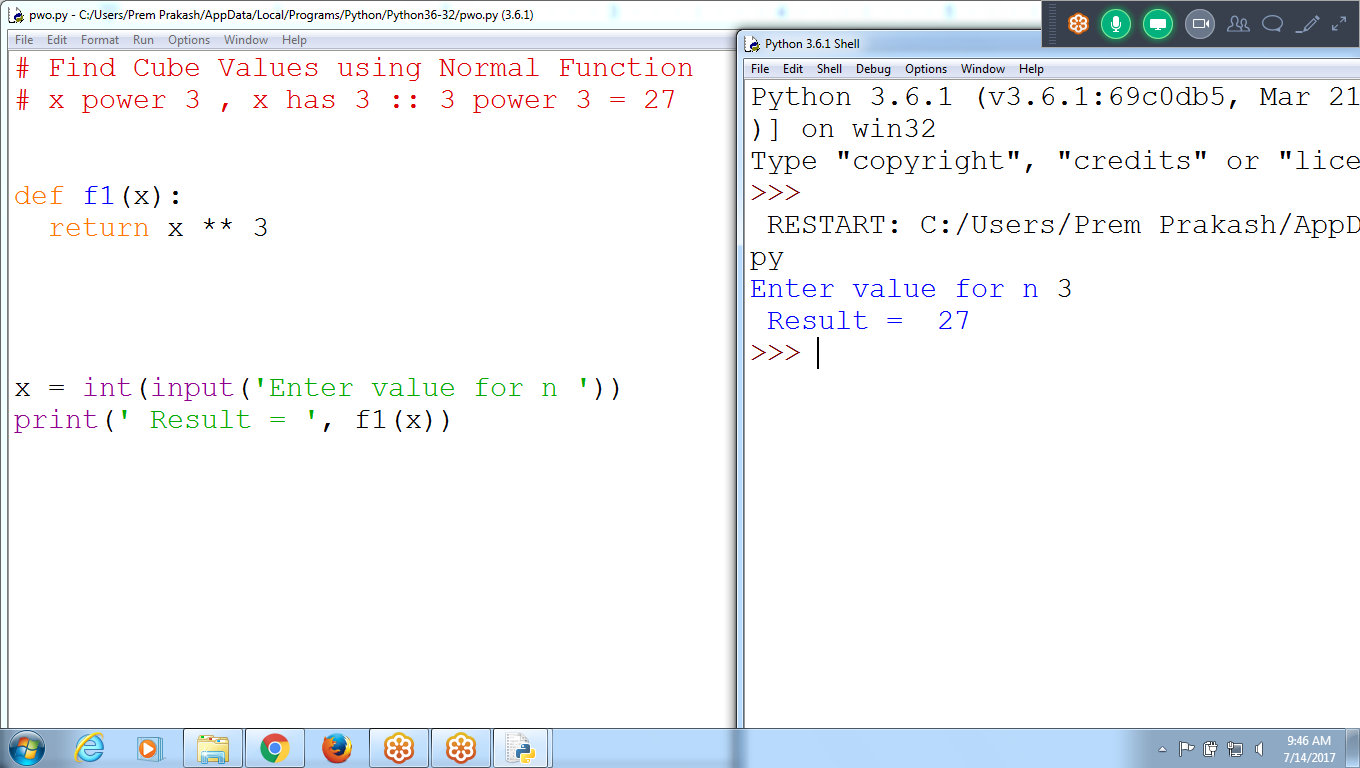
**#Difference between a normal function and a LAMBDA function**

**Normal Function : To Find CUbe values**

****

**# Find Cube Values using Normal Function**

**# x power 3 , x has 3 :: 3 power 3 = 27**

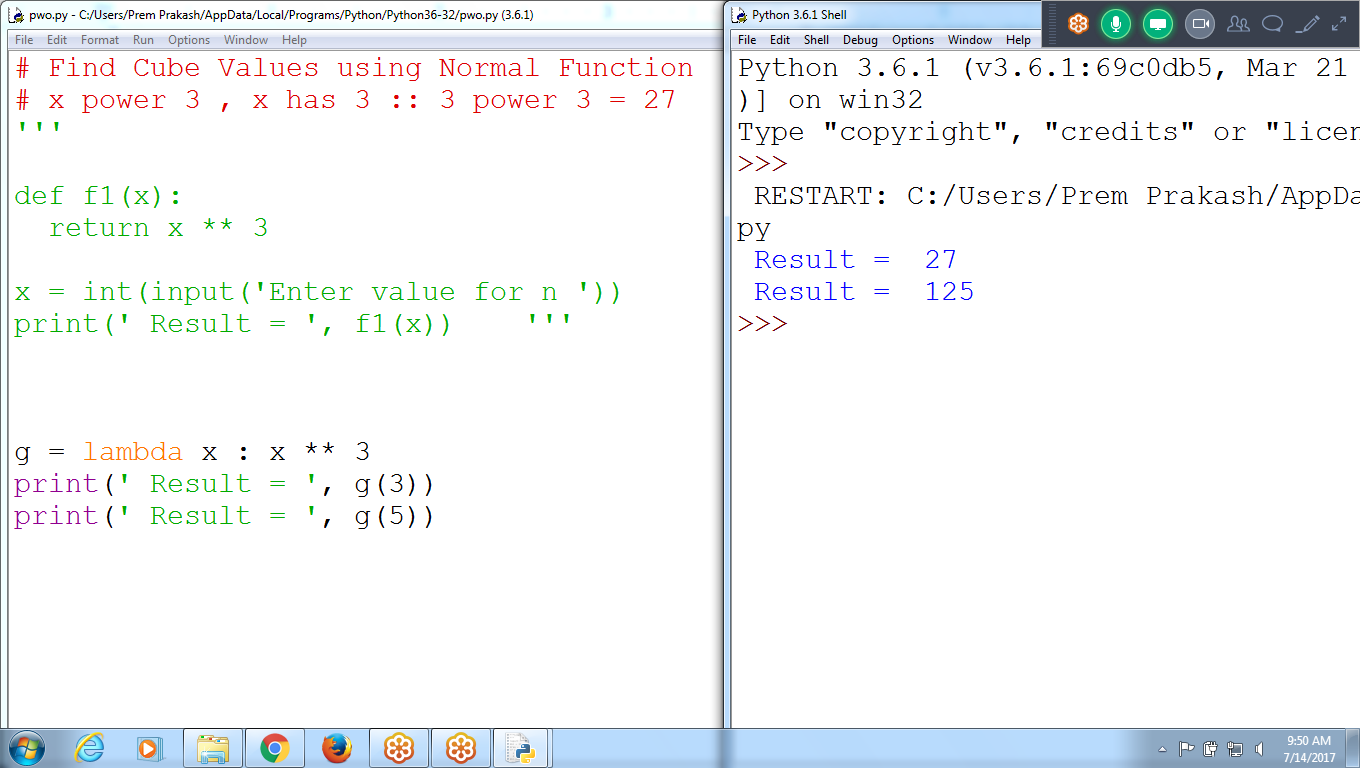
**def f1(x):**

**return x \*\* 3 # X exponent 3**

**x = int(input('Enter value for n '))**

**print(' Result = ', f1(x))**

**Using Lambda Find Power values**

****

**# Find Cube Values using Normal Function**

**# x power 3 , x has 3 :: 3 power 3 = 27**

**'''**

**def f1(x):**

**return x \*\* 3**

**x = int(input('Enter value for n '))**

**print(' Result = ', f1(x)) '''**

**g = lambda x : x \*\* 3**

**print(' Result = ', g(3))**

**print(' Result = ', g(5))**

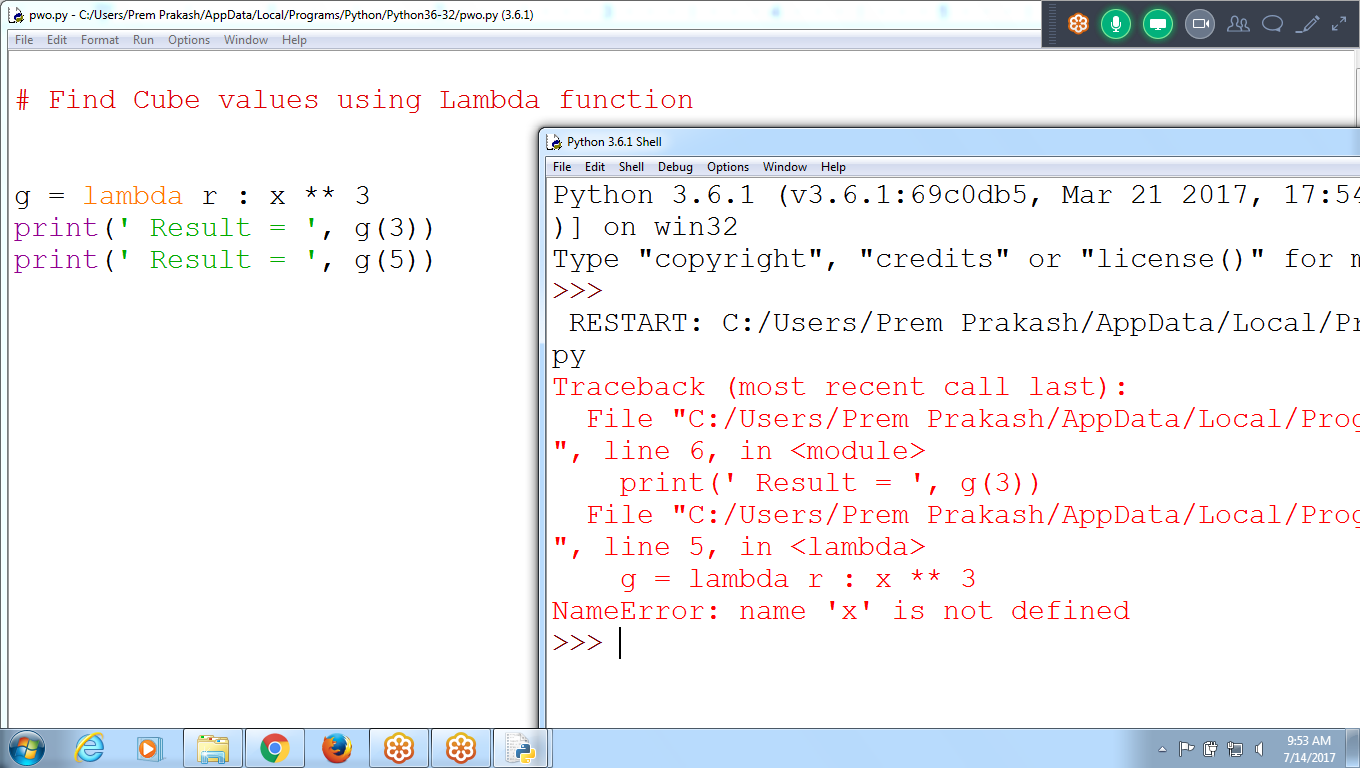
**Input and output variables can not different**

**# Find Cube values using Lambda function**

**g = lambda r : x \*\* 3**

**print(' Result = ', g(3))**

**print(' Result = ', g(5))**

****

# Find CUBE using NORMAL Function

def f1(x):

return x \*\* 3

print(' Cube using Normal Function ', f1(3))

**g = lambda x : x \*\* 3**

print(' Cube Using Lambda = ', g(3))

**# Don't give different variable names for input and output**

#g = lambda r : x \*\* 3

#print(' Cube Using Lambda = ', g(3))

print(' Cube using Lambda = ', lambda r : x \*\* 3)

In Python, anonymous function is a [function](https://www.programiz.com/python-programming/function) that is defined without a name.

While normal **functions are defined using the def keyword,** in Python

**anonymous functions are defined using the lambda keyword.**

Hence, anonymous functions are also called lambda functions.

## **Lambda Functions**

A lambda function has the following syntax.

### **Syntax of Lambda Function**

**lambda arguments: expression**Lambda functions can have **any number of arguments but only one expression.** The expression is evaluated and returned.

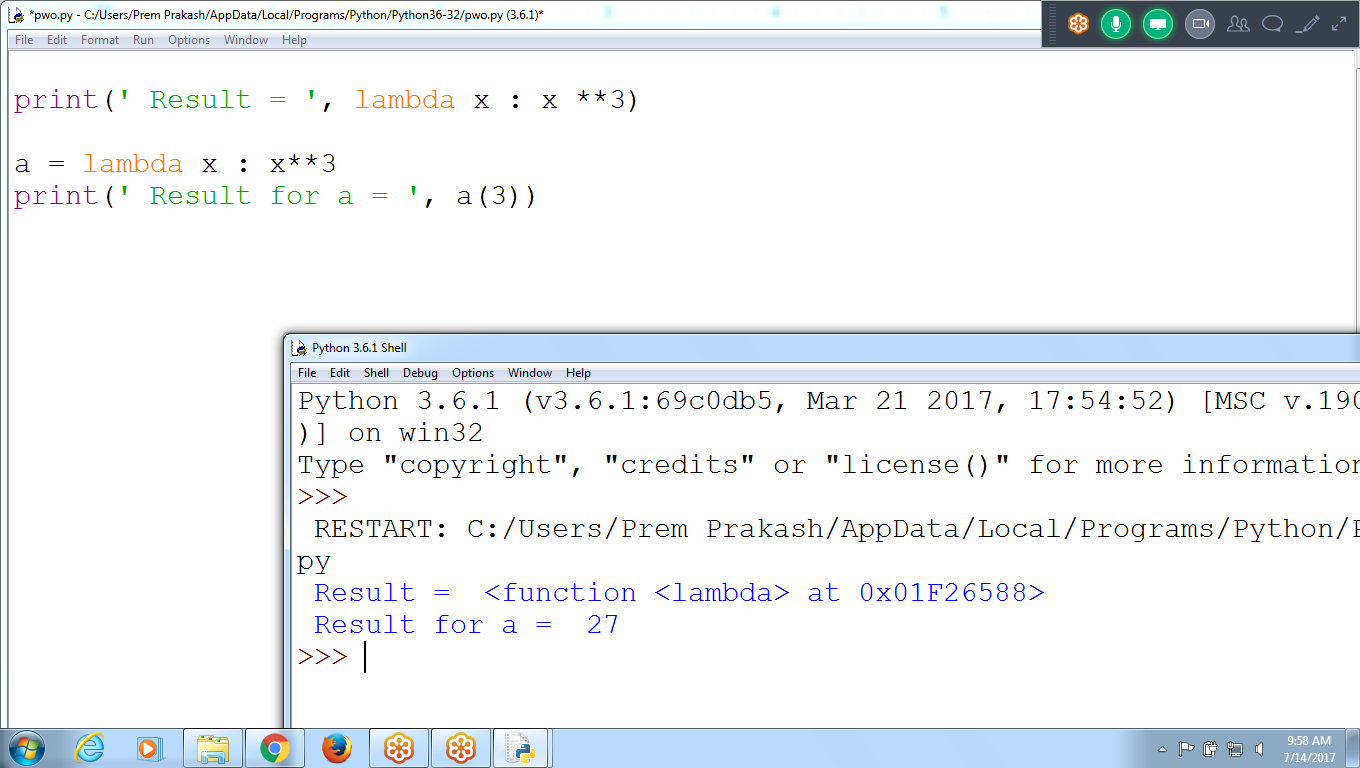
Lambda functions can be used wherever function objects are required.

Lambda Examples 1:

print(' Result = ', lambda x : x \*\*3)

a = lambda x : x\*\*3

print(' Result for a = ', a(3))



Lambda example 2: Passing two parameters

#using Normal Function

def f1(x,y):

return x \*\*3 + y\*\*3

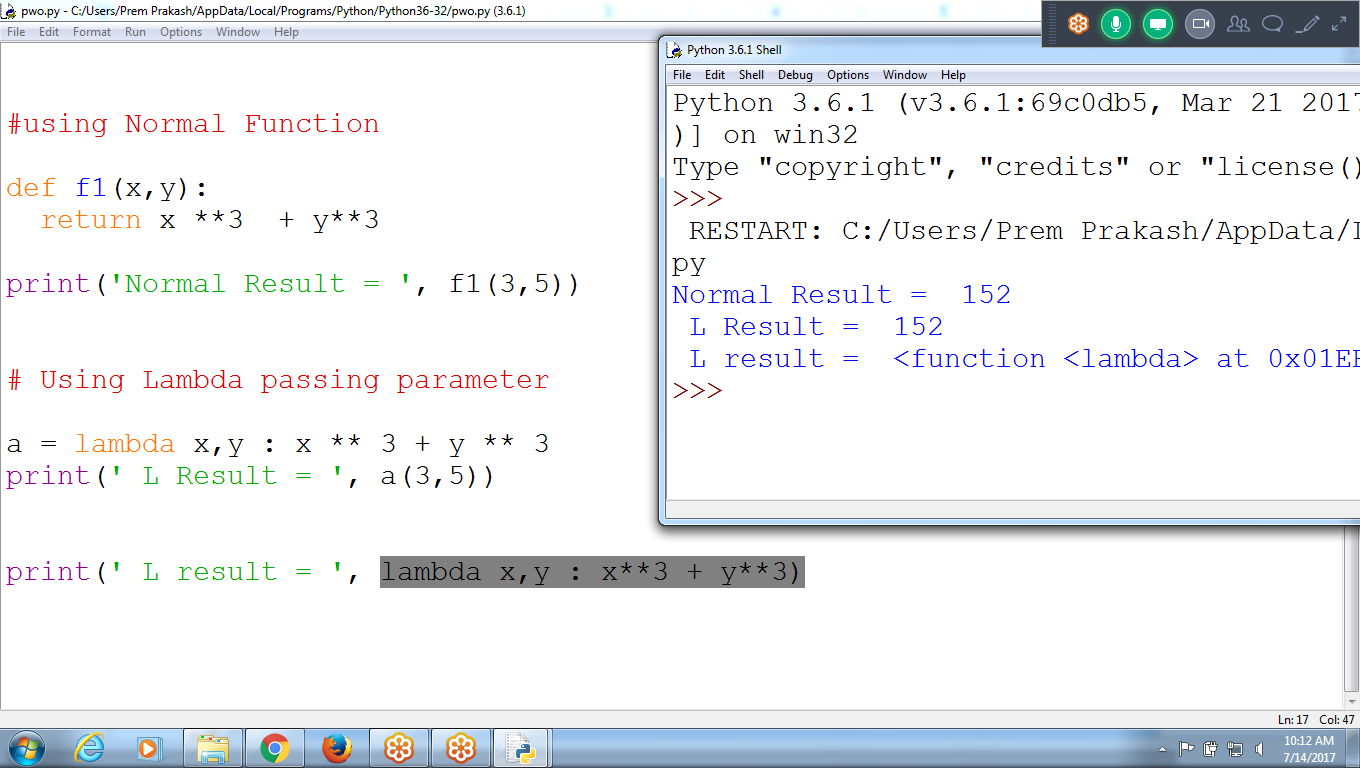
print('Normal Result = ', f1(3,5))

**# Using Lambda passing parameter**

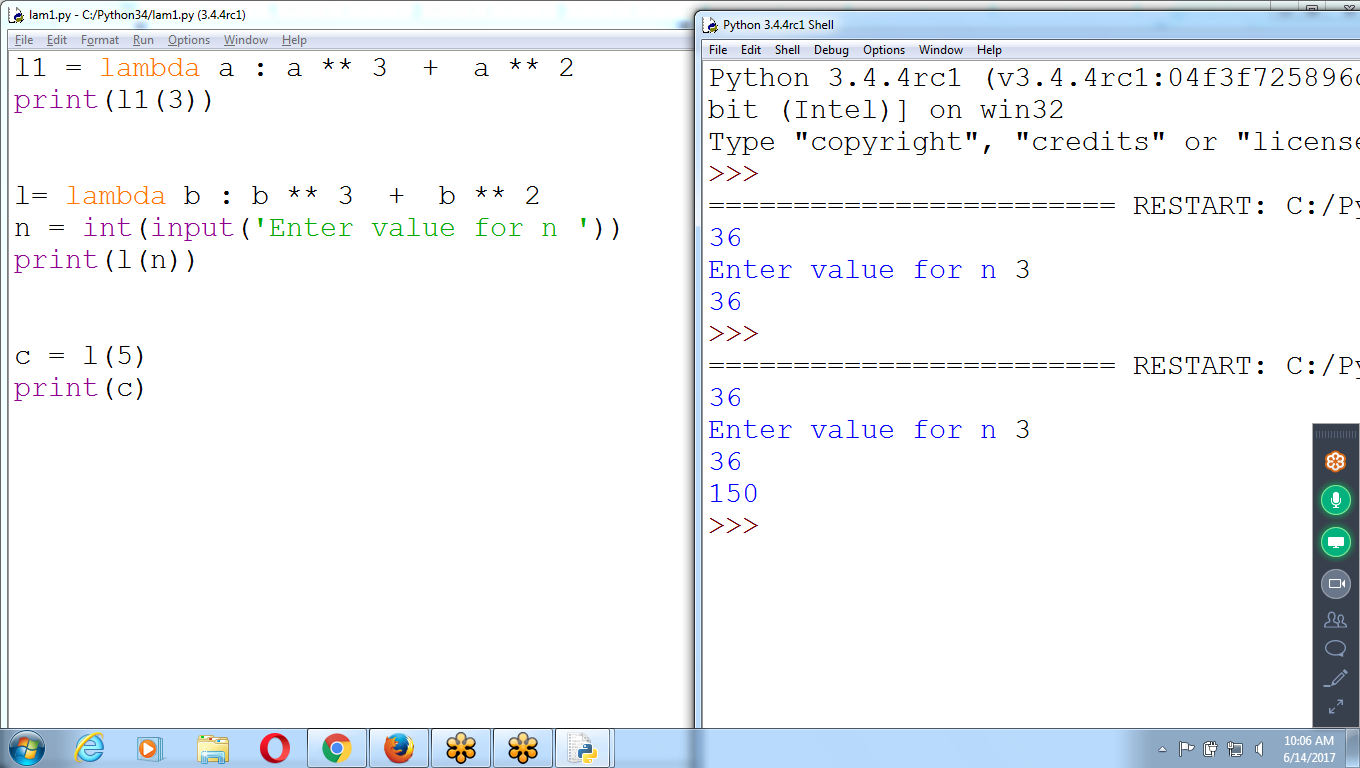
a = lambda **x,y :** x \*\* 3 + y \*\* 3

print(' L Result = ', a(3,5))

print(' L result = ', lambda x,y : x\*\*3 + y\*\*3)



Lambda Examples::



l1 = lambda a : a \*\* 3 + a \*\* 2

print(l1(3))

l= lambda b : b \*\* 3 + b \*\* 2

n = int(input('Enter value for n '))

print(l(n))

c = l(5)

print(c)

### **Use of Lambda Function**

We use lambda functions when we require a nameless function for a short period of time.

In Python, we generally use it as an argument to a higher-order function (a function that takes in other functions as [arguments](https://www.programiz.com/python-programming/function-argument)). Lambda functions are used along with built-in functions like filter(), map() etc.

**Example using LAMBDA Passing Integer and returning Boolean value**

**# Find given number multiples of 4**

**# f1() receiving integer and returning Boolean type**

**def f1(n):**

**if( n % 4 == 0):**

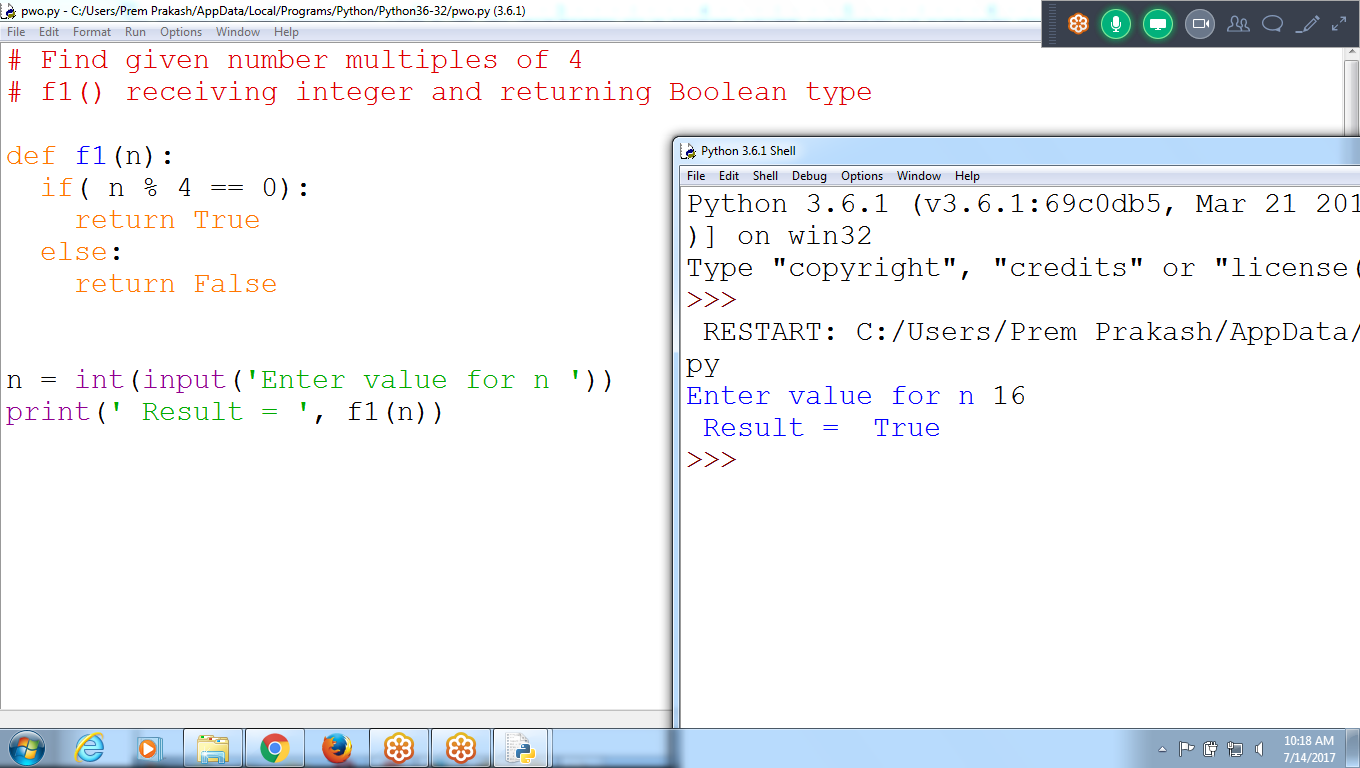
**return True**

**else:**

**return False**

**n = int(input('Enter value for n '))**

**print(' Result = ', f1(n))**

****

**Using Lambda find Multiples of 4**

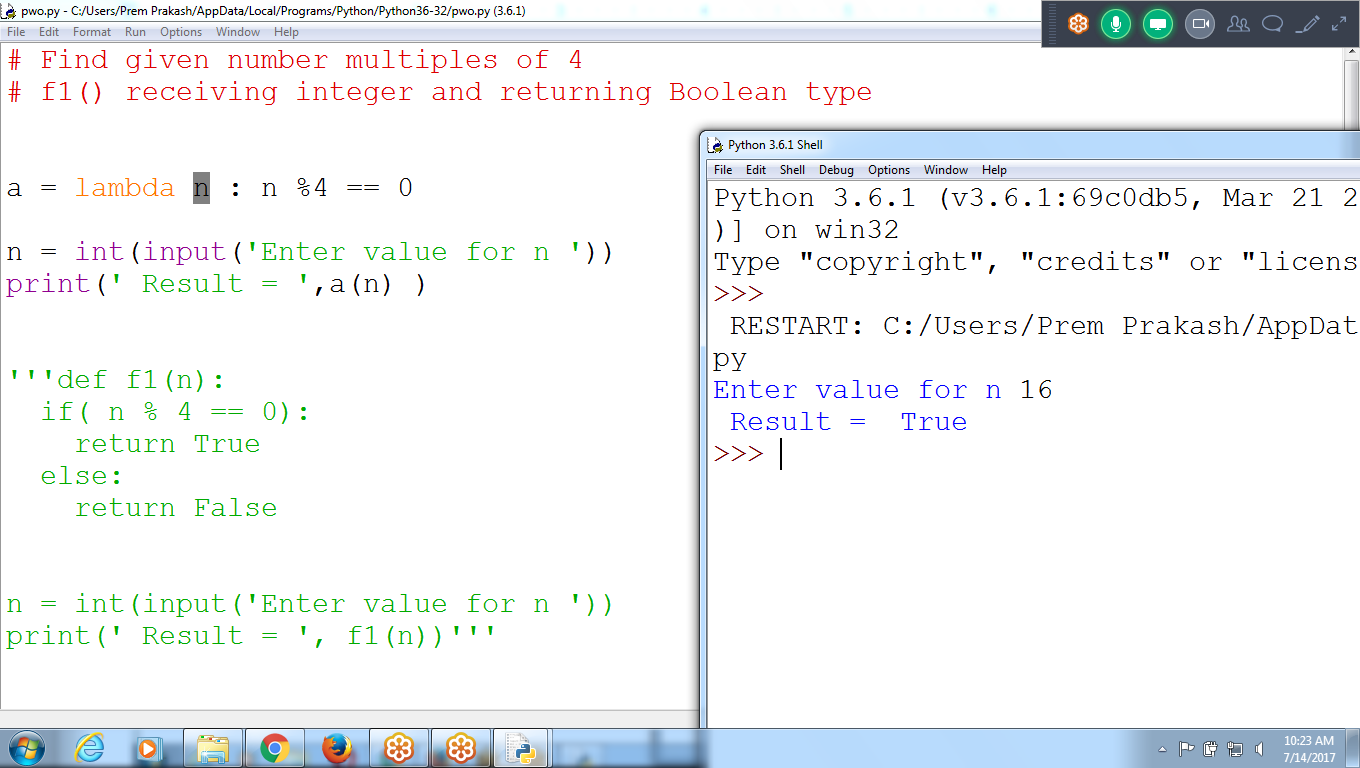
**# Find given number multiples of 4**

**# f1() receiving integer and returning Boolean type**

**a = lambda n : n %4 == 0**

**n = int(input('Enter value for n '))**

**print(' Result = ',a(n) )**

****

# Program to accept number, and find divisible by 4 or not

l = lambda x : x%4 == 0

n = int(input('Enter value for n '))

print(l(n))

**Return List of Elements which are divisible by 2 in a given list using NORMAL Function**

**# Input as list of numbers**

**# Result is list of number divisible by 2**

**def f1(ls):**

**rls = []**

**for l in ls:**

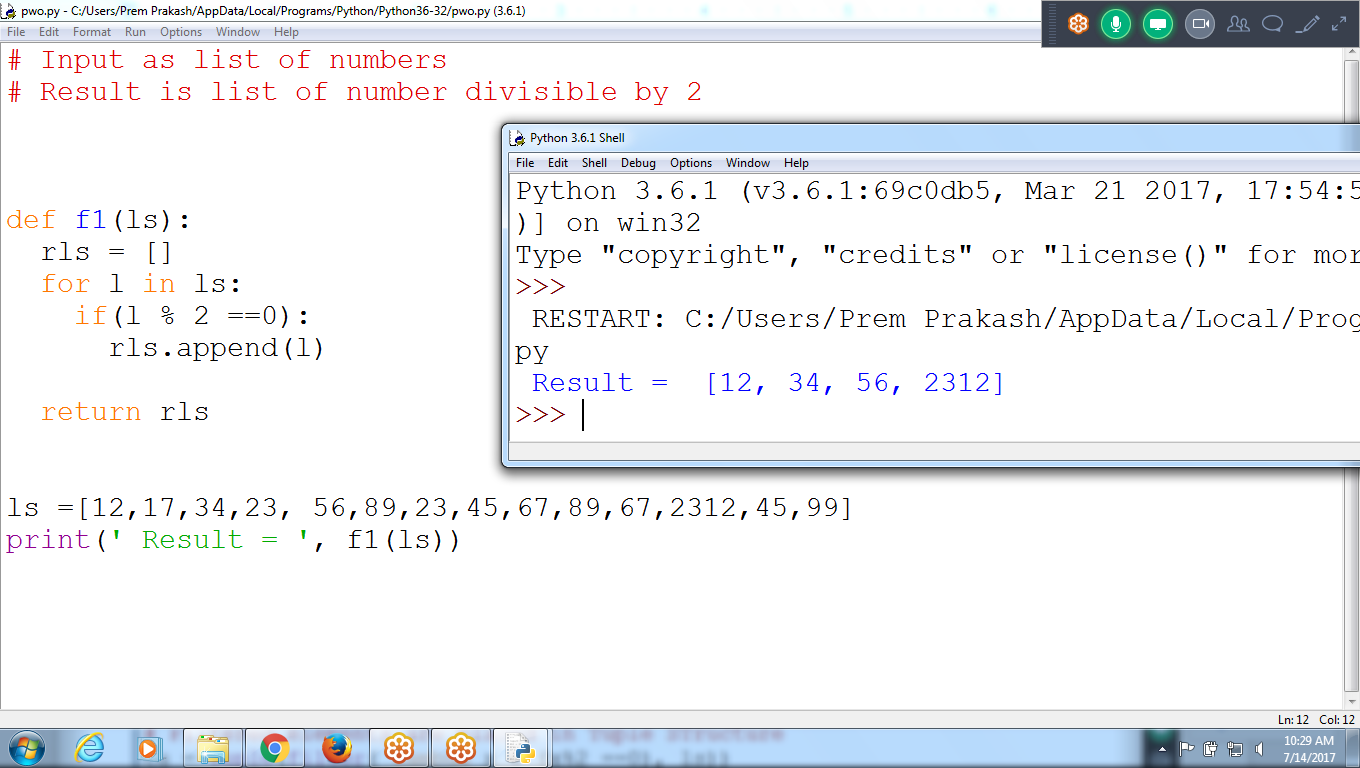
**if(l % 2 ==0):**

**rls.append(l)**

**return rls**

**ls =[12,17,34,23, 56,89,23,45,67,89,67,2312,45,99]**

**print(' Result = ', f1(ls))**

****