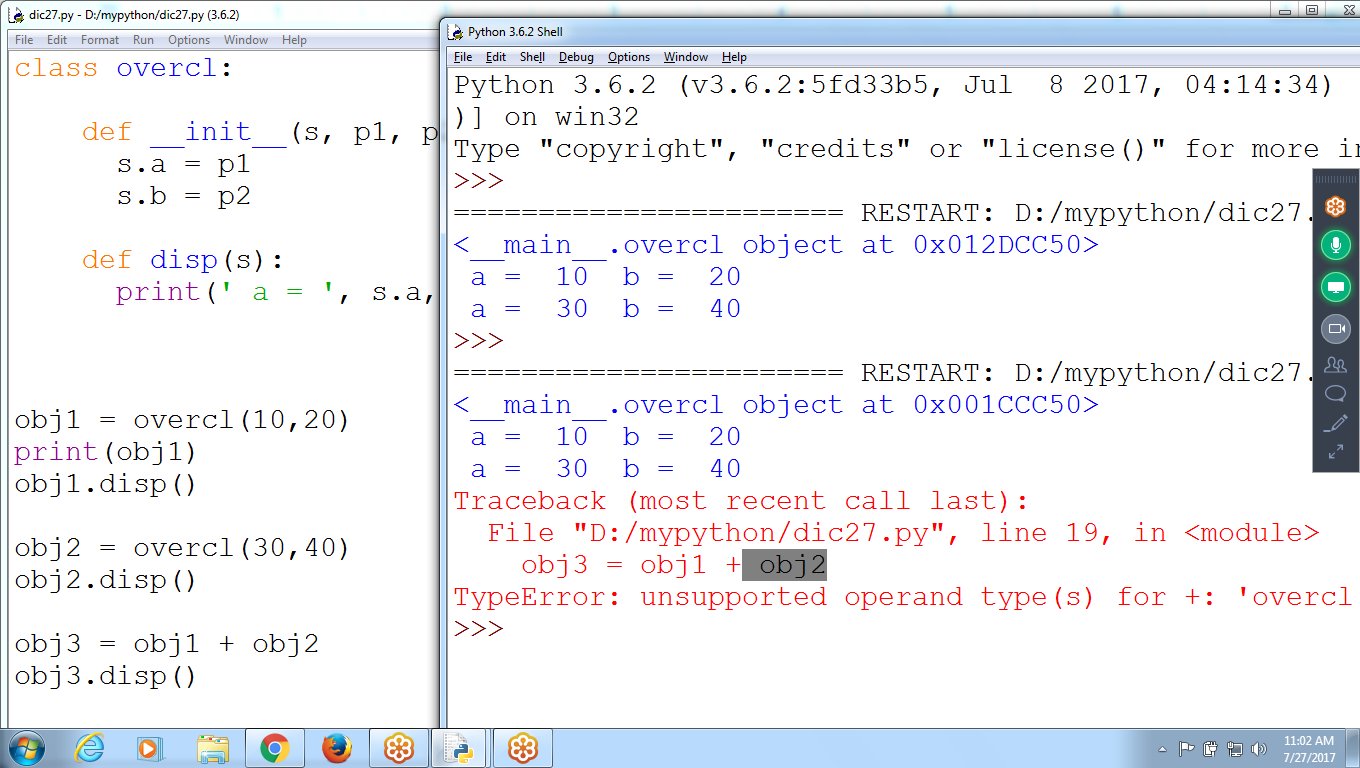
**OBject Addition**

**Overloading Operators on Objects**

* **: Works for integer, float, list and String variable**
* **: How to WorK with OBJeCTS**

****

**# Object Addition**

**class overcl:**

**def \_\_init\_\_(s, p1, p2):**

**s.a = p1**

**s.b = p2**

**def disp(s):**

**print(' a = ', s.a, ' b = ', s.b)**

**obj1 = overcl(10,20)**

**print(obj1)**

**obj1.disp()**

**obj2 = overcl(30,40)**

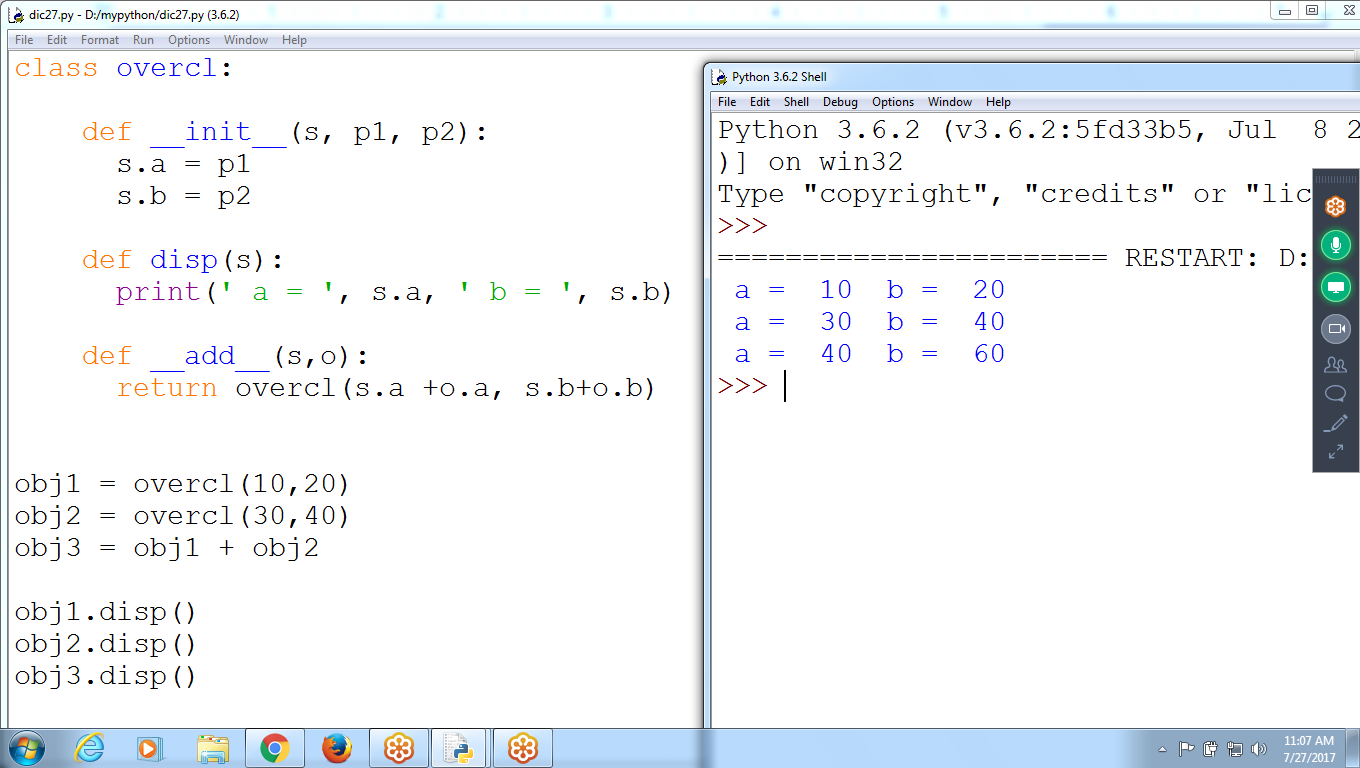
**obj2.disp()**

**obj3 = obj1 + obj2**

**obj3.disp()**

**Can Perform Addition (+) on Integer, float, string**

**Then how to Perform on OBjects**

****

**# Object Addition**

**class overcl:**

**def \_\_init\_\_(s, p1, p2):**

**s.a = p1**

**s.b = p2**

**def disp(s):**

**print(' a = ', s.a, ' b = ', s.b)**

**def \_\_add\_\_(s,o):**

**return overcl(s.a +o.a, s.b+o.b)**

**obj1 = overcl(10,20)**

**obj2 = overcl(30,40)**

**obj3 = obj1 + obj2**

**obj1.disp()**

**obj2.disp()**

**obj3.disp()**

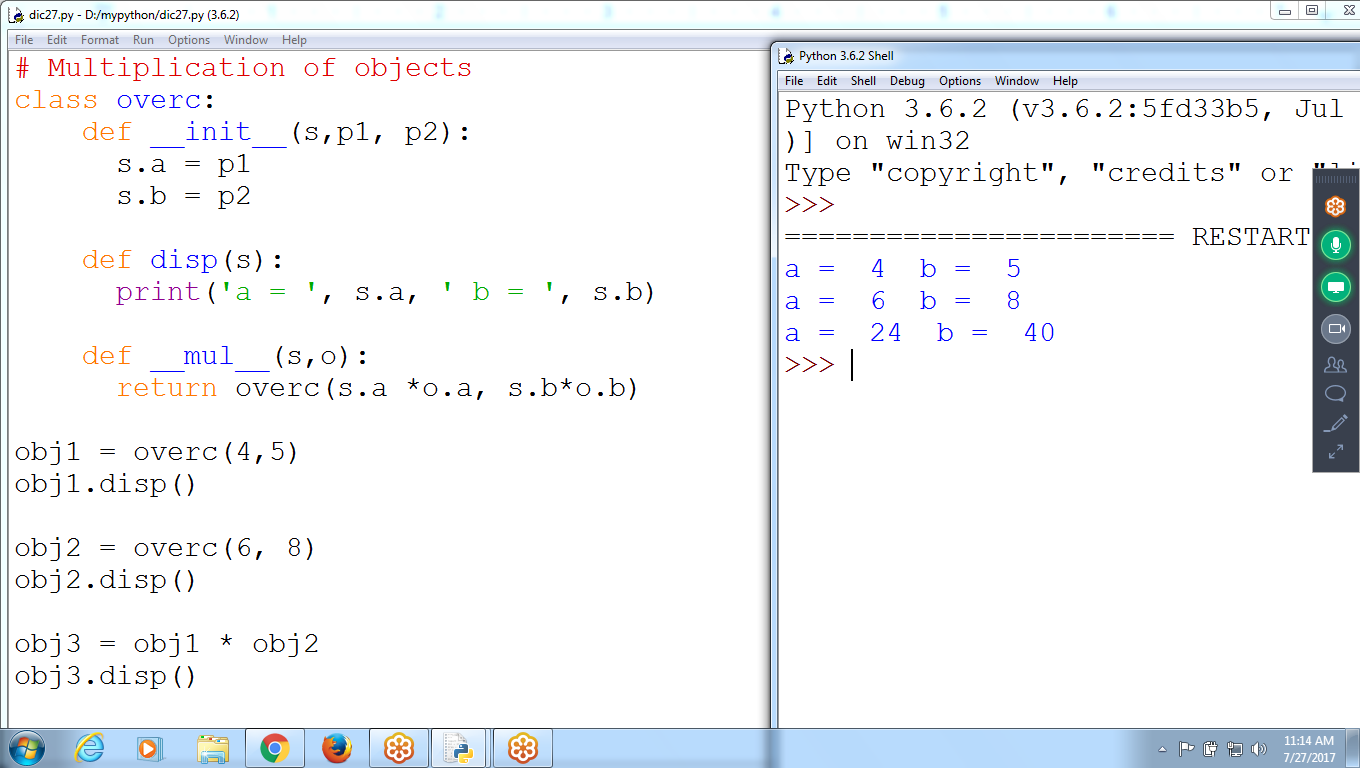
**Sum of Multiple Objects**

**obj3 = obj1 + obj2 ( Two Objects)**

**Try Logic::**

**Obj = ( ( ( obj1 + obj2) + obj3 ) + obj4 )**

**Objects Multiplication**

****

# Multiplication of objects

class overc:

def \_\_init\_\_(s,p1, p2):

s.a = p1

s.b = p2

def disp(s):

print('a = ', s.a, ' b = ', s.b)

def \_\_mul\_\_(s,o):

return overc(s.a \*o.a, s.b\*o.b)

obj1 = overc(4,5)

obj1.disp()

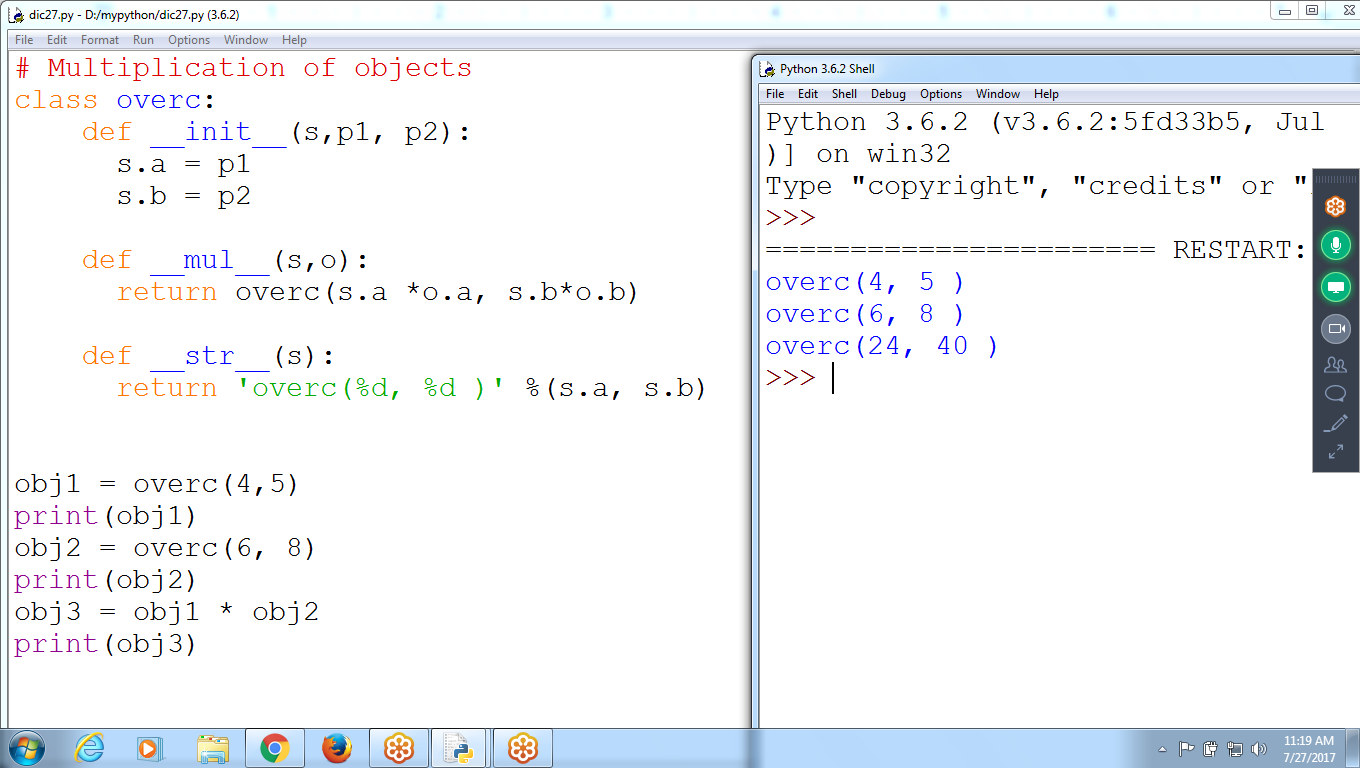
obj2 = overc(6, 8)

obj2.disp()

obj3 = obj1 \* obj2

obj3.disp()

**Overloading Print() using \_\_str\_\_**

****

**# Multiplication of objects**

**class overc:**

**def \_\_init\_\_(s,p1, p2):**

**s.a = p1**

**s.b = p2**

**def \_\_mul\_\_(s,o):**

**return overc(s.a \*o.a, s.b\*o.b)**

**def \_\_str\_\_(s):**

**return 'overc(%d, %d )' %(s.a, s.b)**

**obj1 = overc(4,5)**

**print(obj1)**

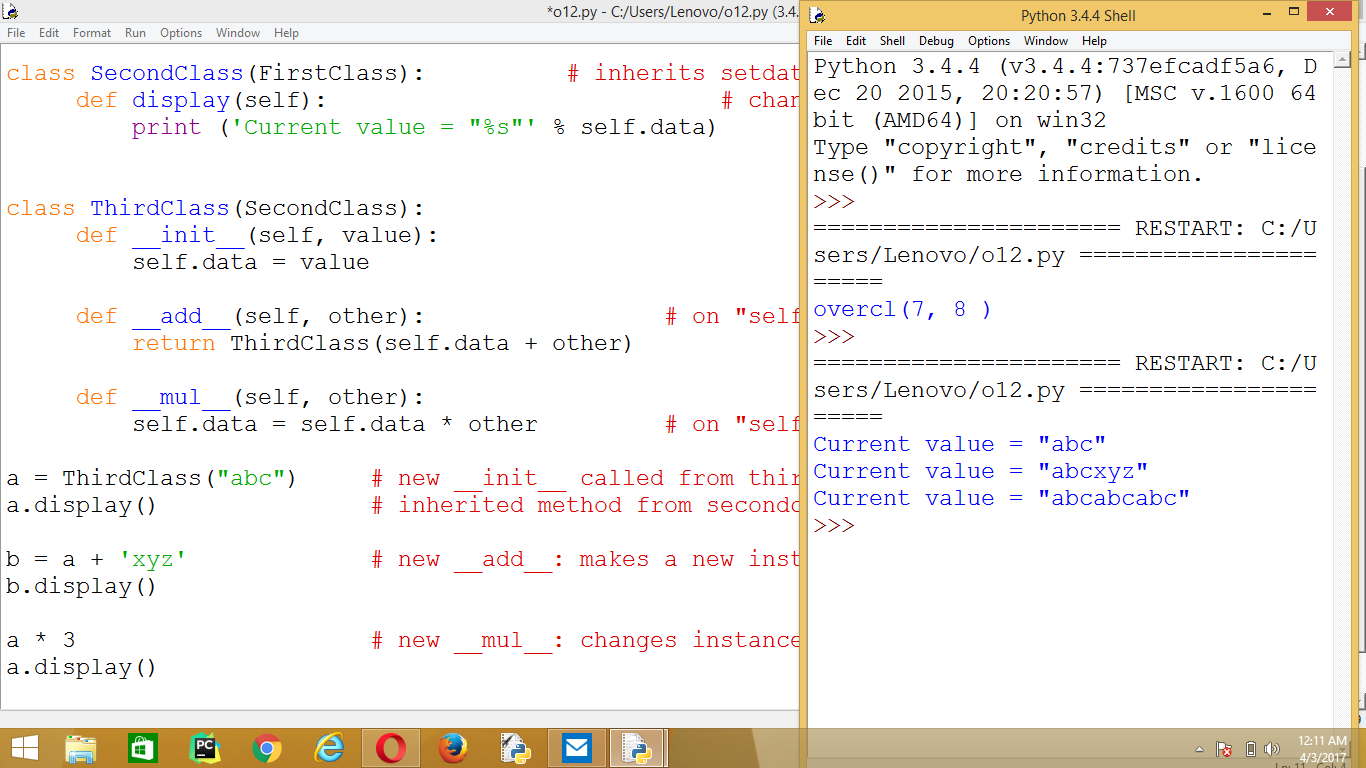
**obj2 = overc(6, 8)**

**print(obj2)**

**obj3 = obj1 \* obj2**

**print(obj3)**

**Using \_\_add\_\_ , \_\_mul\_\_**

****

class FirstClass:

def setdata(self, value):

self.data = value

def display(self):

print (self.data )

class SecondClass(FirstClass): # inherits setdata

def display(self): # changes display

print ('Current value = "%s"' % self.data)

class ThirdClass(SecondClass):

def \_\_init\_\_(self, value):

self.data = value

def **\_\_add\_\_**(self, other): # on "self + other"

return ThirdClass(self.data + other)

def \_\_mul\_\_(self, other):

self.data = self.data \* other # on "self \* other"

a = ThirdClass("abc") # new \_\_init\_\_ called from thirdcl

a.display() # inherited method from secondclass

b = a + 'xyz' # new \_\_add\_\_: makes a new instance

b.display()

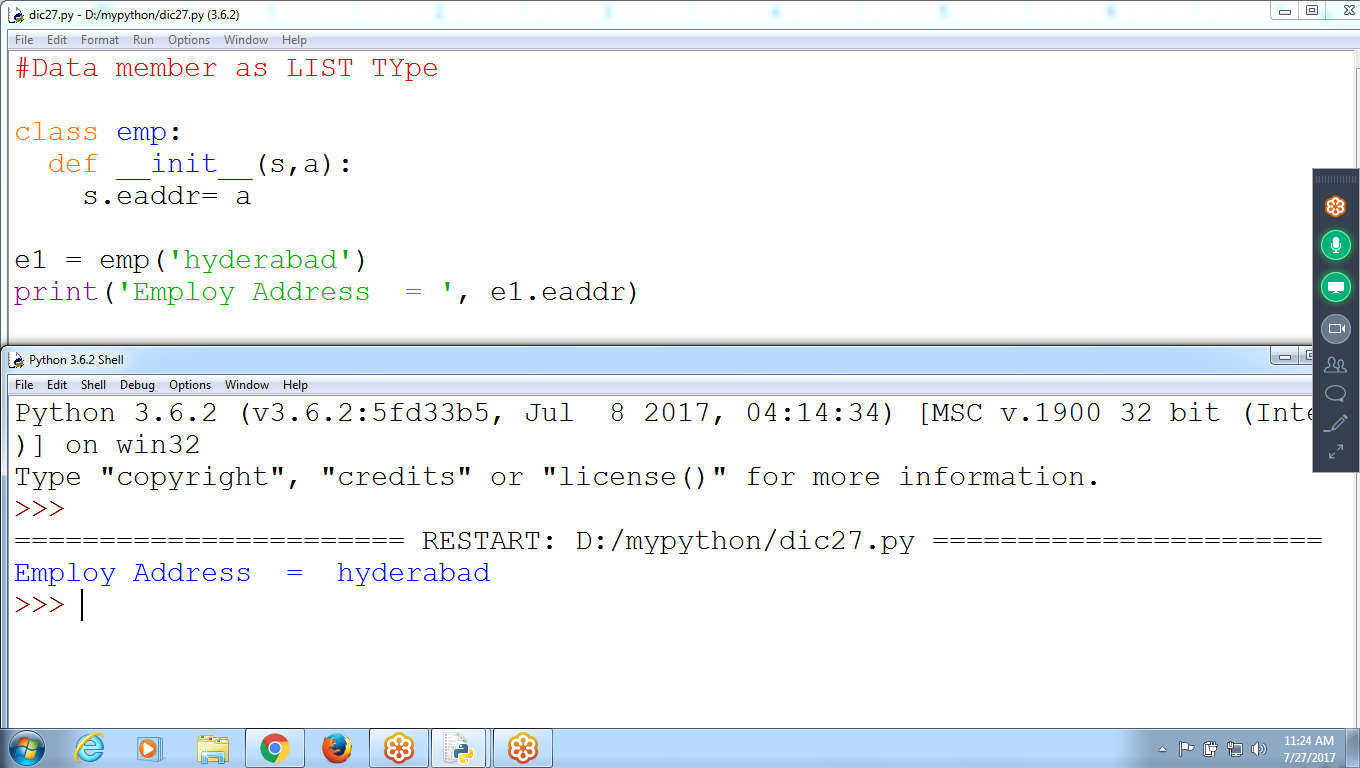
a \* 3 # new \_\_mul\_\_: changes instance in-place

a.display()

**Data Member as STRING Type**

**Current Emp Object address is taking as String,**

**where i can store one address value**

****

**#Data member as LIST TYpe**

**class emp:**

**def \_\_init\_\_(s,a):**

**s.eaddr= a**

**e1 = emp('hyderabad')**

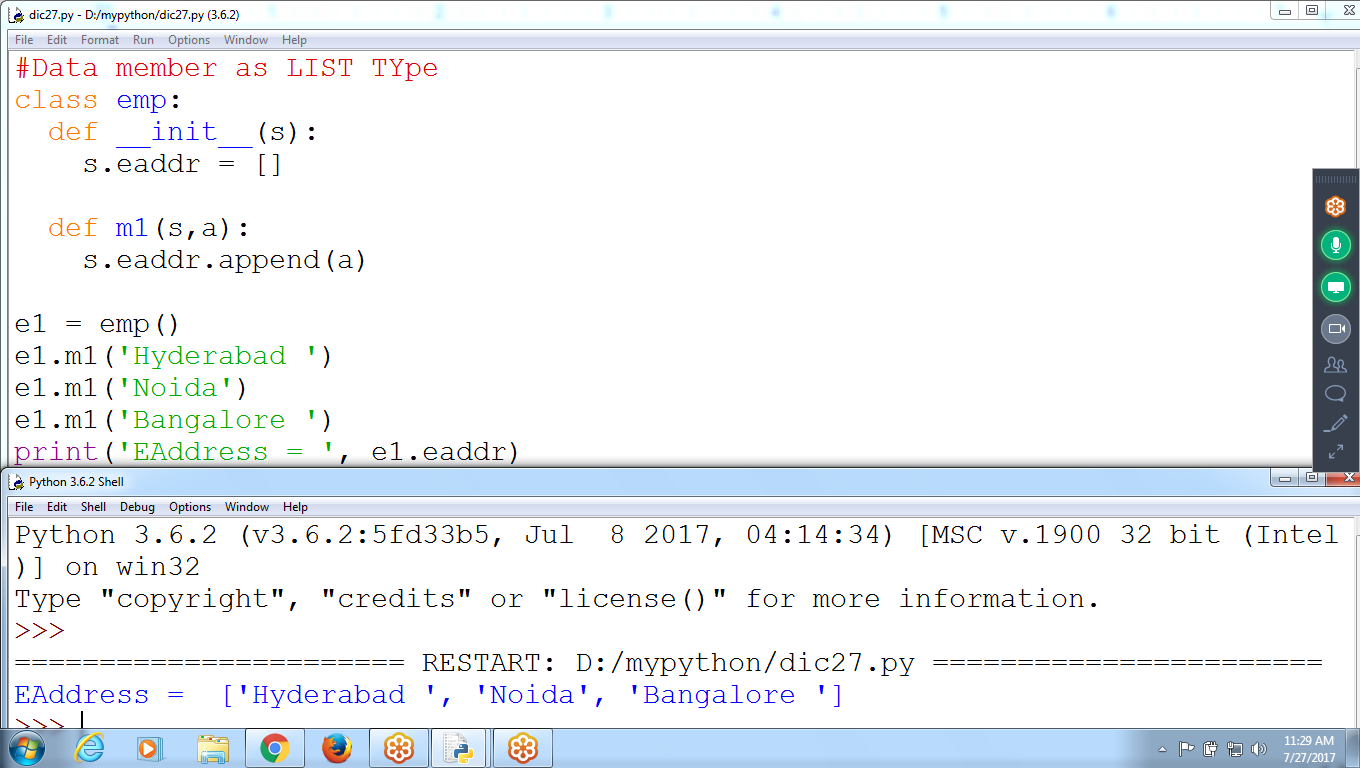
**print('Employ Address = ', e1.eaddr)**

**Data Member as LIST TYpe**

**How to Store Multiple Addresses in single emp object**

**Data Member as LIST Data Type**

**# eid is datamember of type LIST**



# emp class contain eid of type LIST TYPE and m1()

#Data member as LIST TYpe

class emp:

def \_\_init\_\_(s):

s.eaddr = []

def m1(s,a):

s.eaddr.append(a)

e1 = emp()

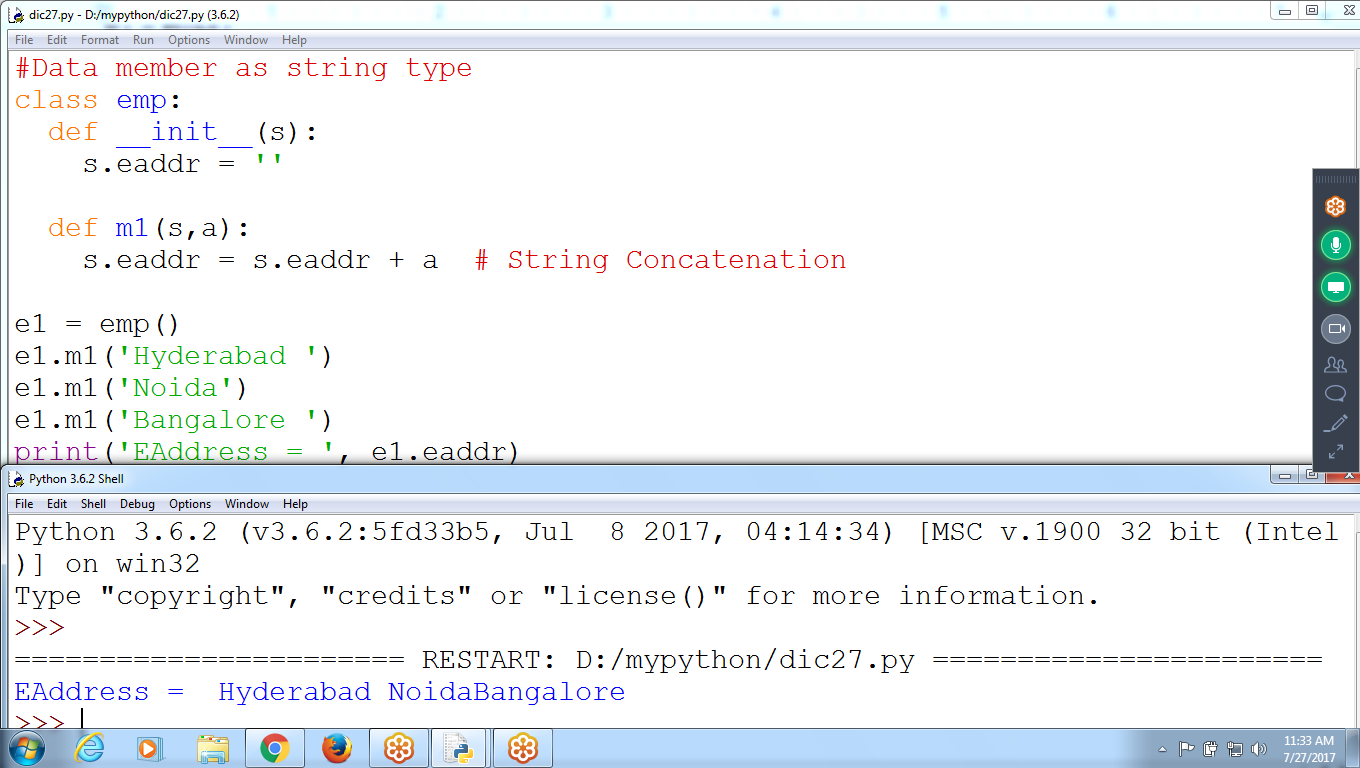
e1.m1('Hyderabad ')

e1.m1('Noida')

e1.m1('Bangalore ')

print('EAddress = ', e1.eaddr)

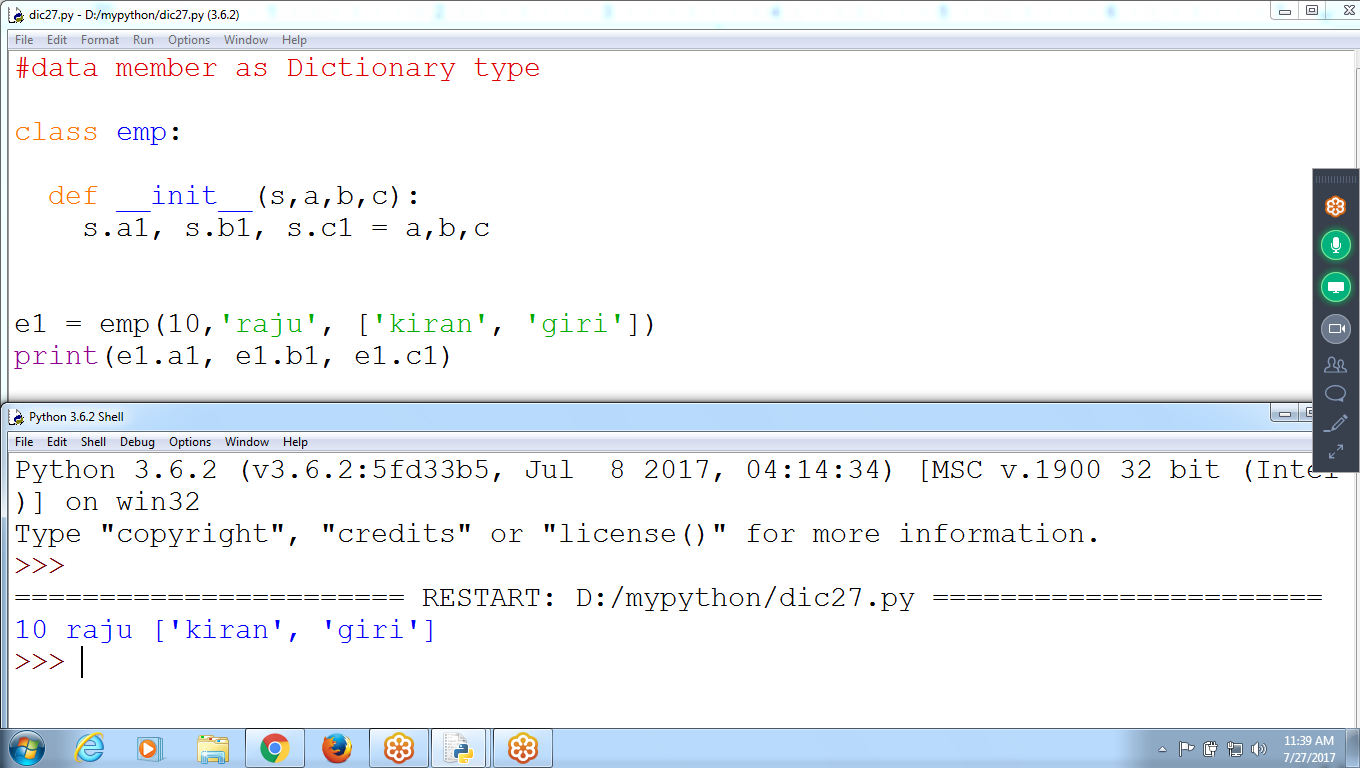
**Datatype as String, storing multiple values using string concatenation**



**Passing 3 different values (integer, string, list) and receiving 3 Parameters (a,b,c)**

E1 object contains 3 Data members

E1.a1, e1.b1, e1.c1



#data member as Dictionary type

class emp:

def \_\_init\_\_(s,a,b,c):

s.a1, s.b1, s.c1 = a,b,c

e1 = emp(10,'raju', ['kiran', 'giri'])

print(e1.a1, e1.b1, e1.c1)

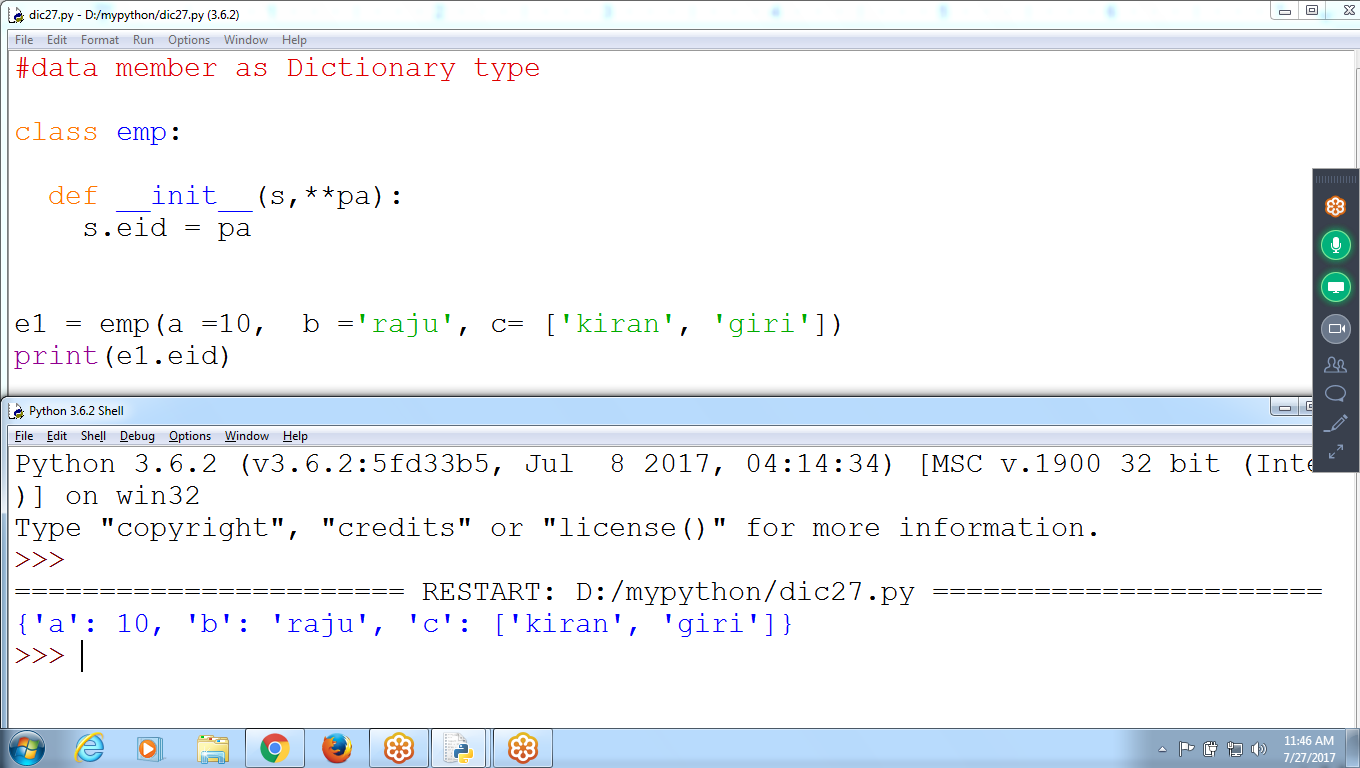
**Pass 3 different values, and receive single parameter as Dictionary**

**Data Member as Dictionary Type**

using Dictionary Variable as Data Member

Eid is dictionary type, \*\*pa represents dictionary,

while passing requires variable and value, where variable is key



#data member as Dictionary type

class emp:

def \_\_init\_\_(s,\*\*pa):

s.eid = pa

e1 = emp(a =10, b ='raju', c= ['kiran', 'giri'])

print(e1.eid)