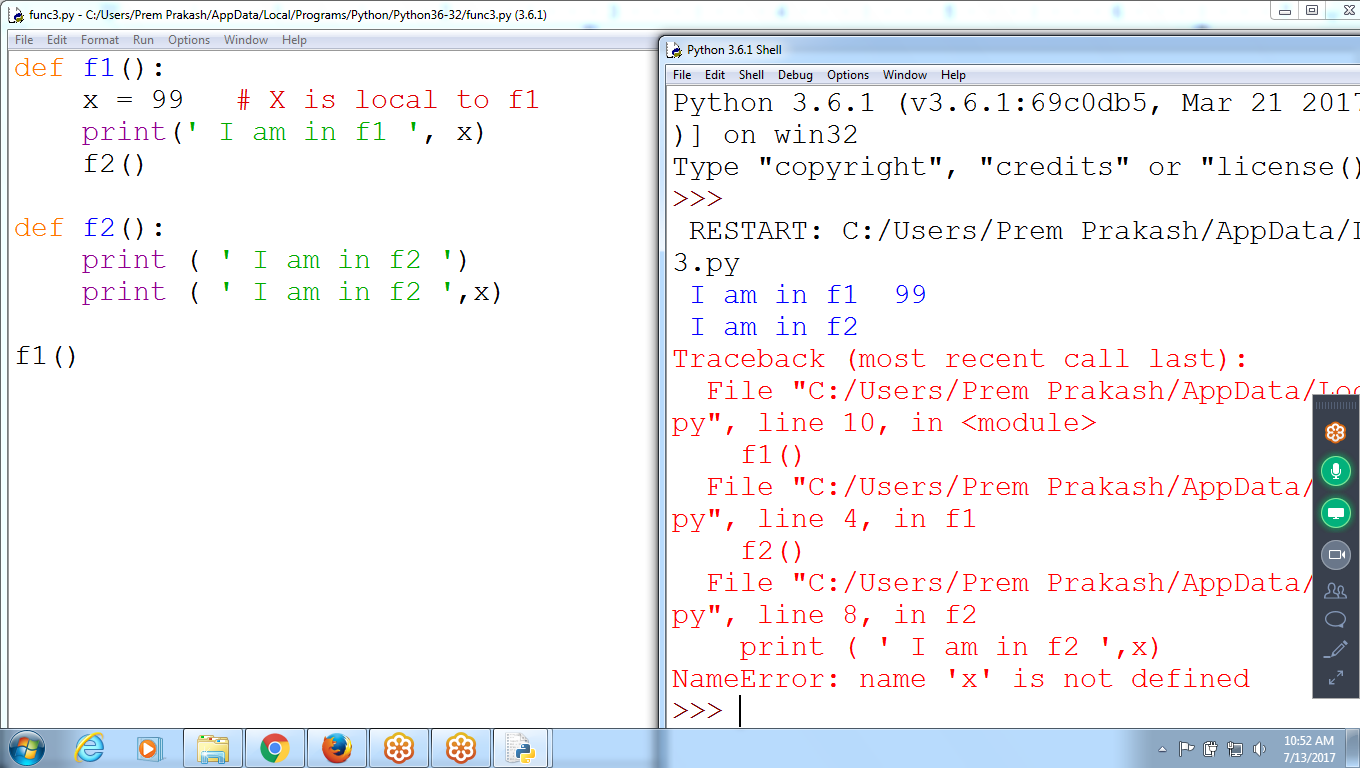
Without Inner Functions



def f1():

x = 99 # X is local to f1

print(' I am in f1 ', x)

f2()

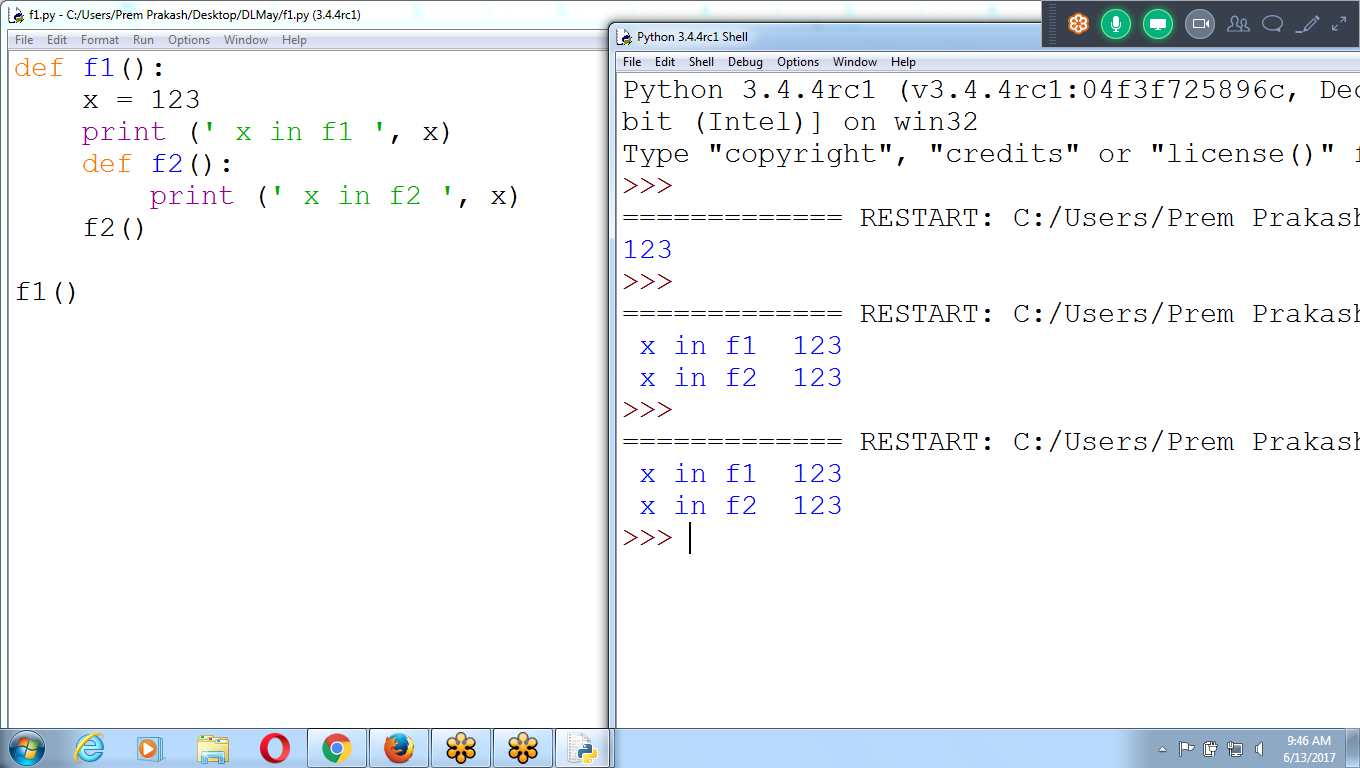
def f2():

print ( ' I am in f2 ')

print ( ' I am in f2 ',x)

f1()

# Inner Function : Defining Function definition, within the function



# Inner Function : Defining Function definition, with in the function

def f1():

x = 123

print (' x in f1 ', x)

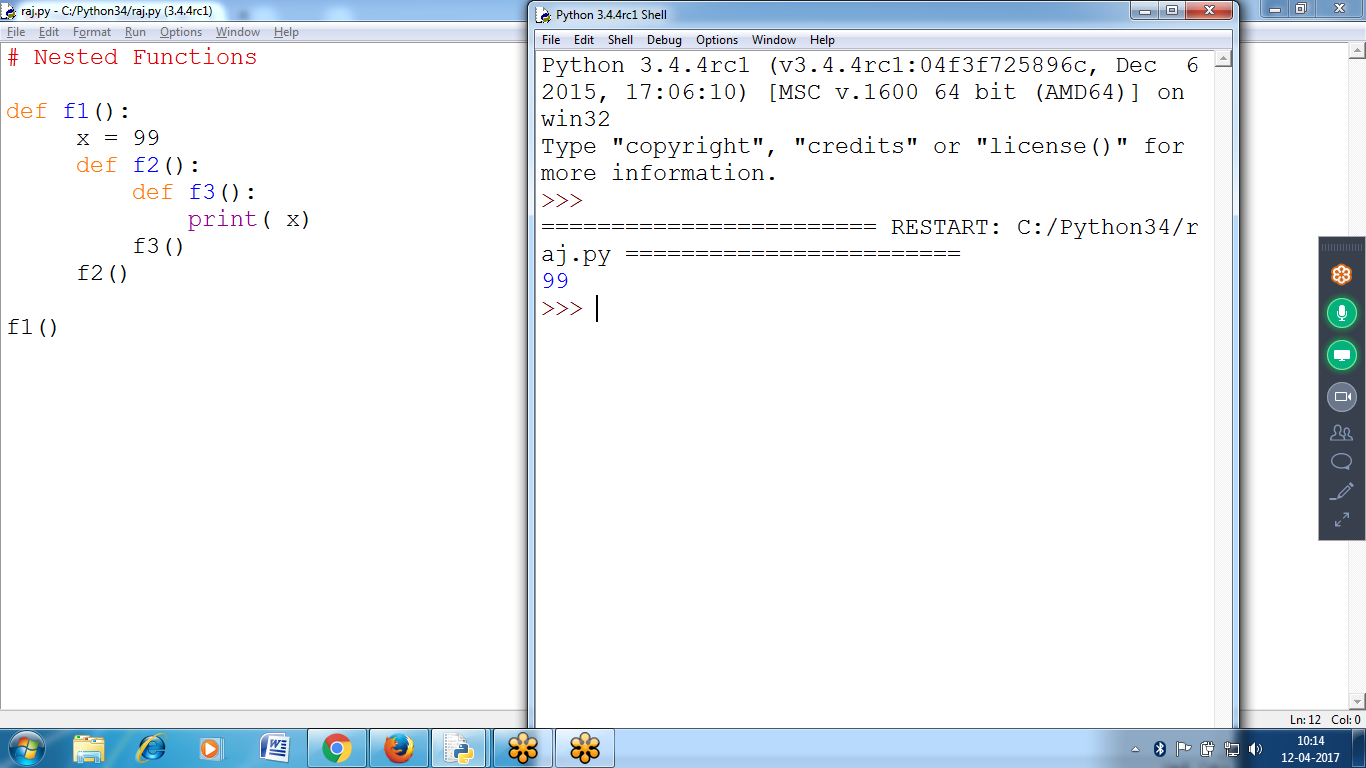
def f2():

print (' x in f2 ', x)

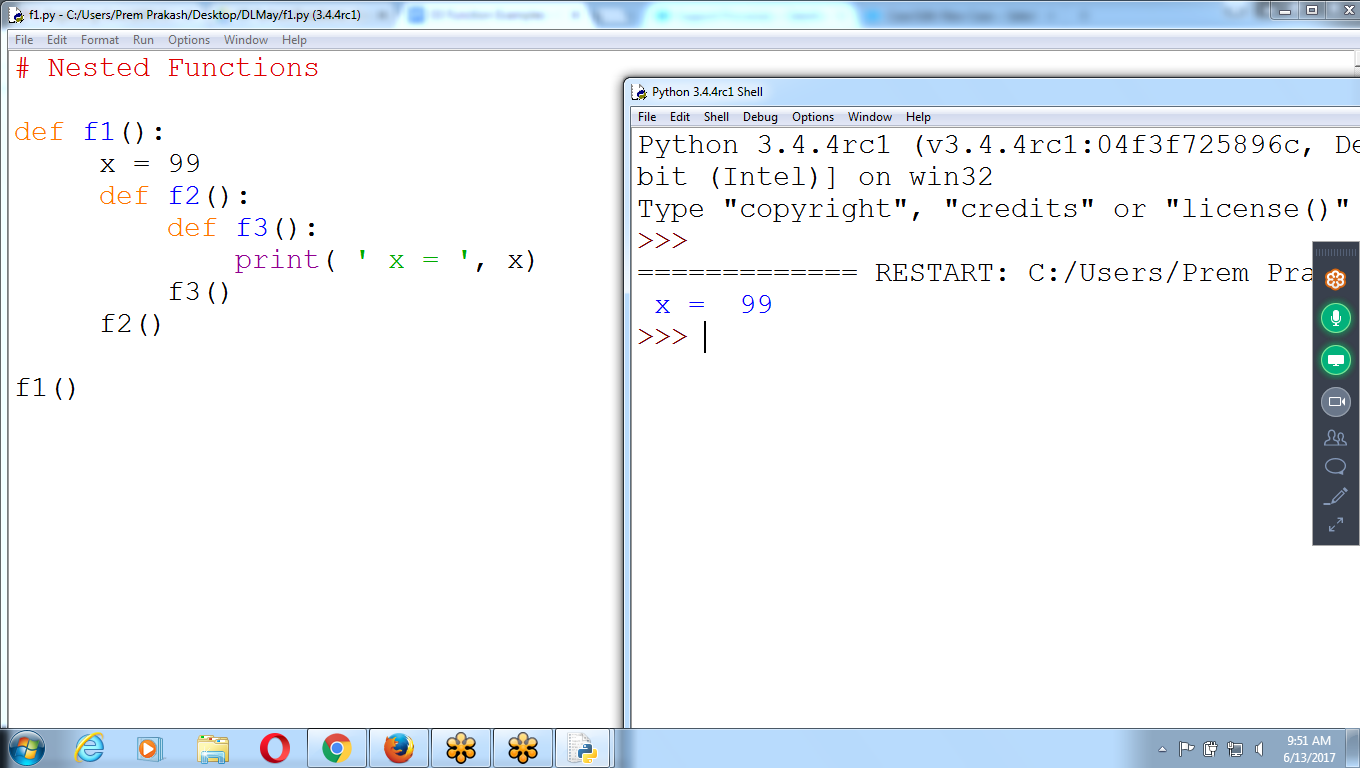
f2()

f1()

Nested Functions



Nested Functions check using Debugger :: Click “STEP”



# Nested Functions

def f1():

x = 99

def f2():

def f3():

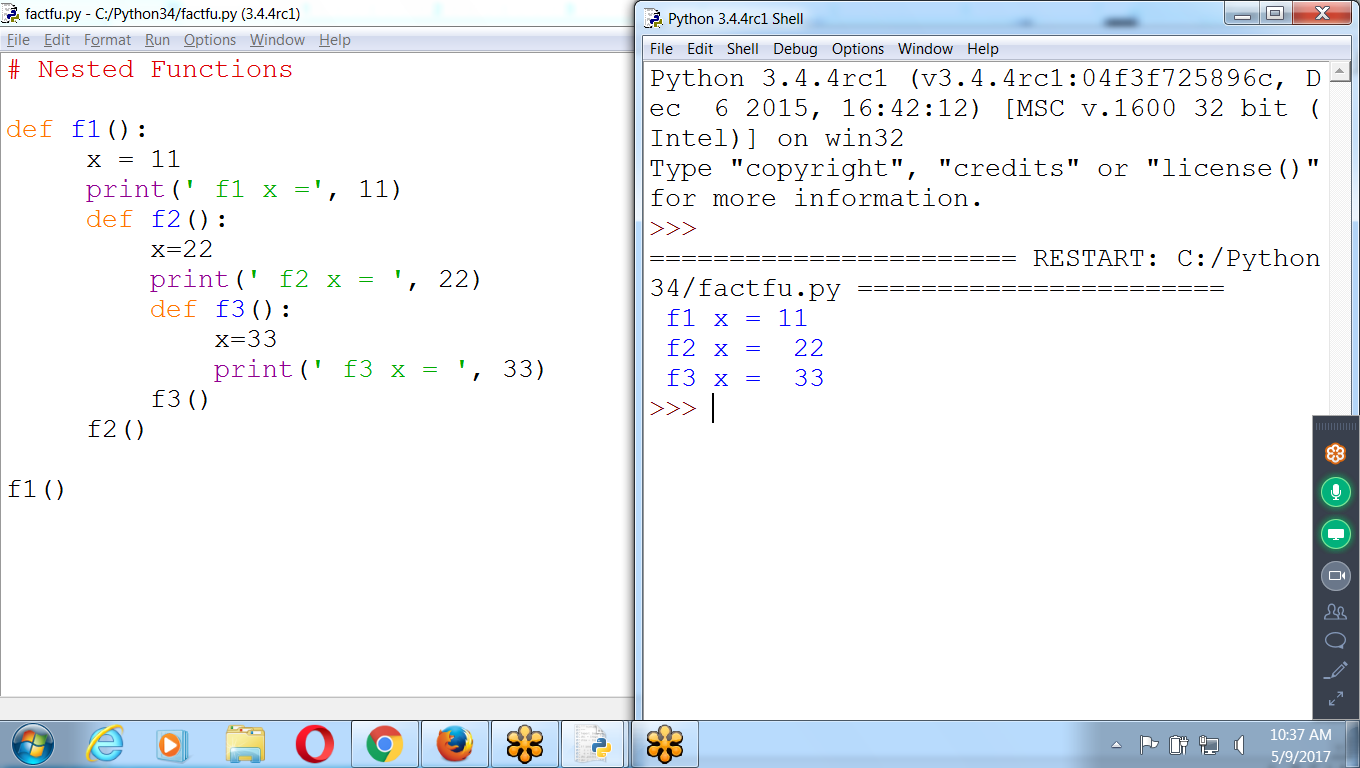
print( ' x = ', x)

f3()

f2()

f1()

Nested Functions



# Nested Functions

def f1():

x = 11

print(' f1 x =', 11)

def f2():

x=22

print(' f2 x = ', 22)

def f3():

x=33

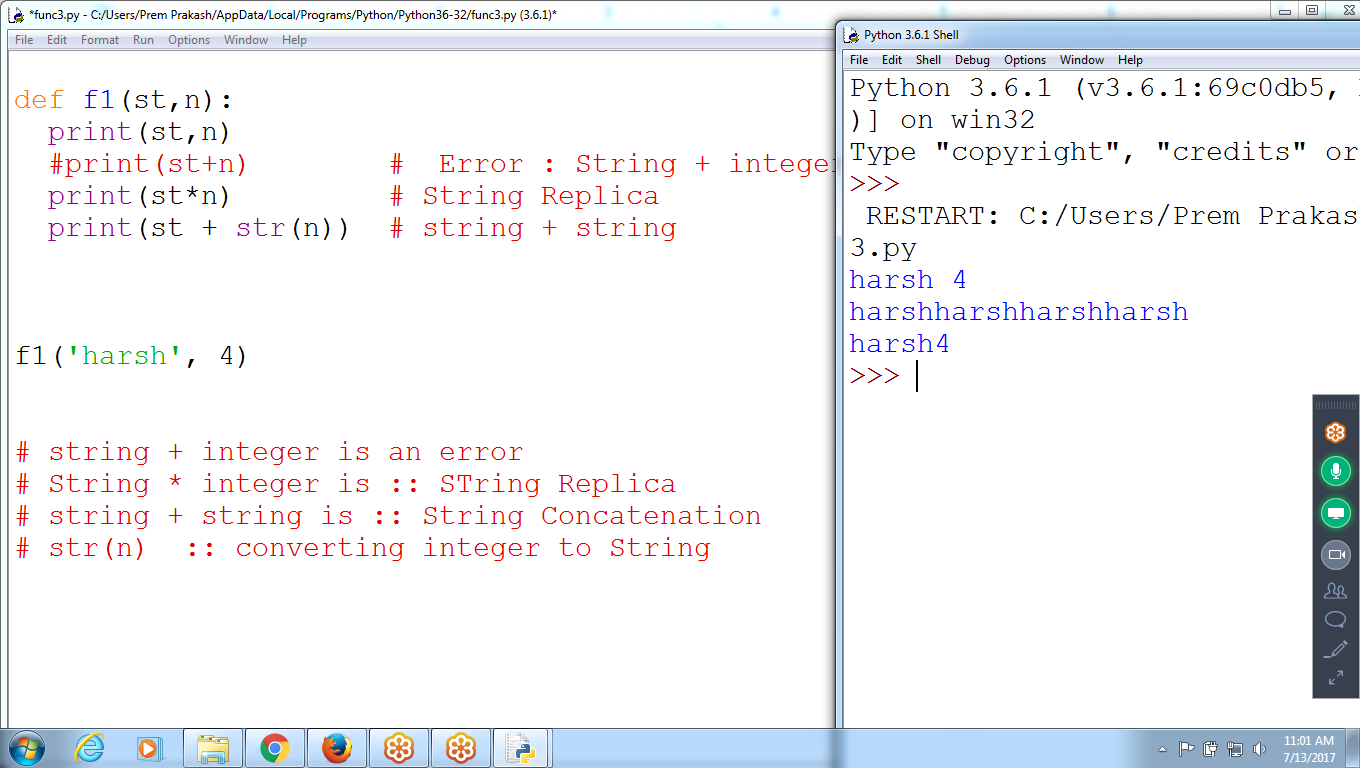
print(' f3 x = ', 33)

f3()

f2()

f1()

**Typeless Functions**



def f1(st,n):

print(st,n)

#print(st+n) # Error : String + integer

print(st\*n) # String Replica

print(st + str(n)) # string + string

f1('harsh', 4)

# string + integer is an error

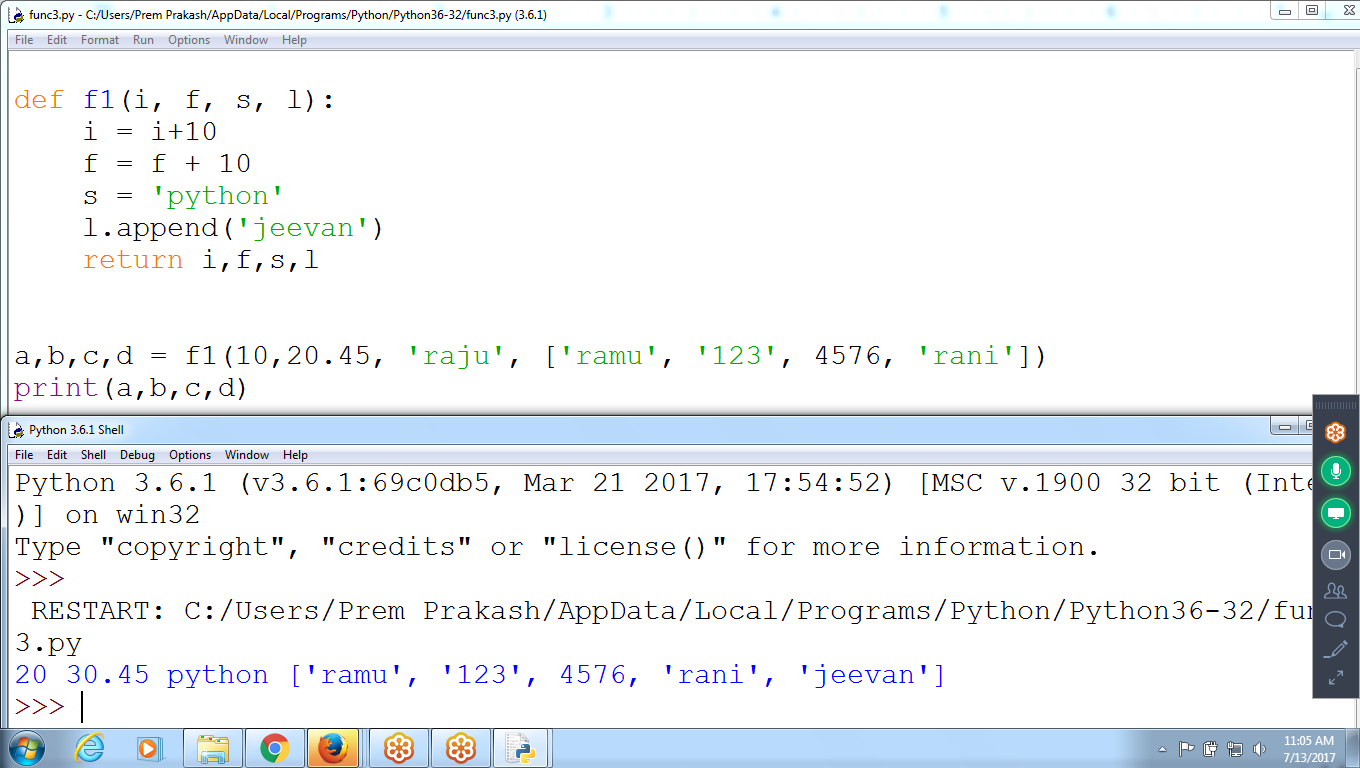
# String \* integer is :: STring Replica

# string + string is :: String Concatenation

# str(n) :: converting integer to String

# Multiple Type Return Values

Integer, float, string and list



def f1(i, f, s, l):

i = i+10

f = f + 10

s = 'python'

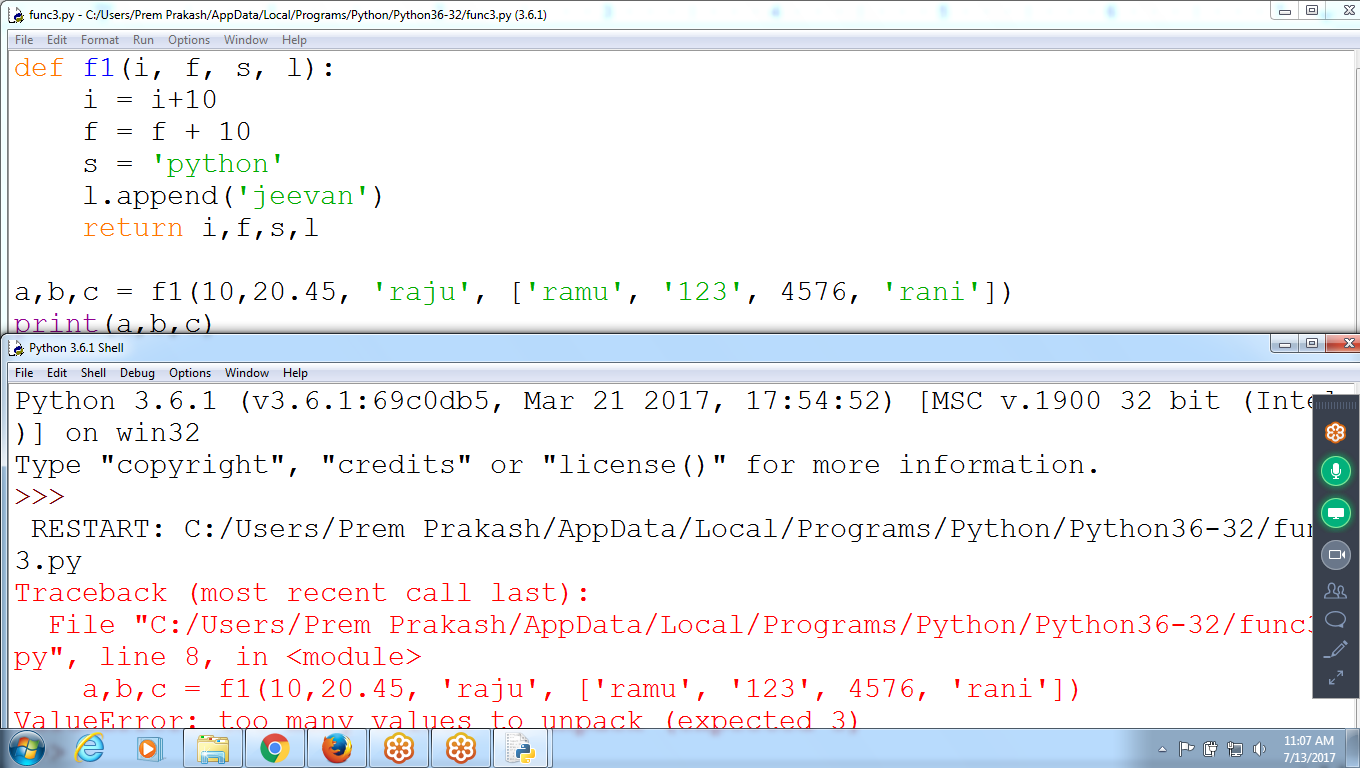
l.append('jeevan')

return i,f,s,l

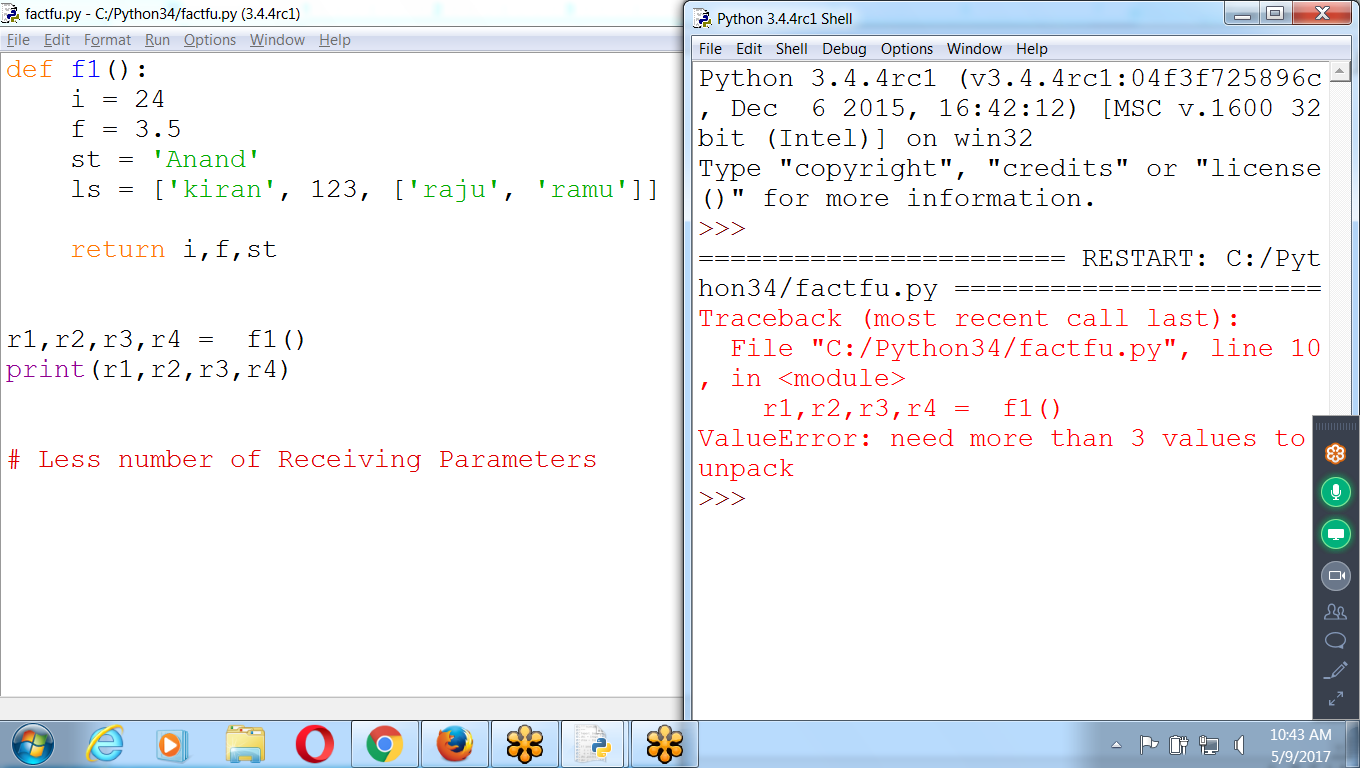
a,b,c,d = f1(10,20.45, 'raju', ['ramu', '123', 4576, 'rani'])

print(a,b,c,d)

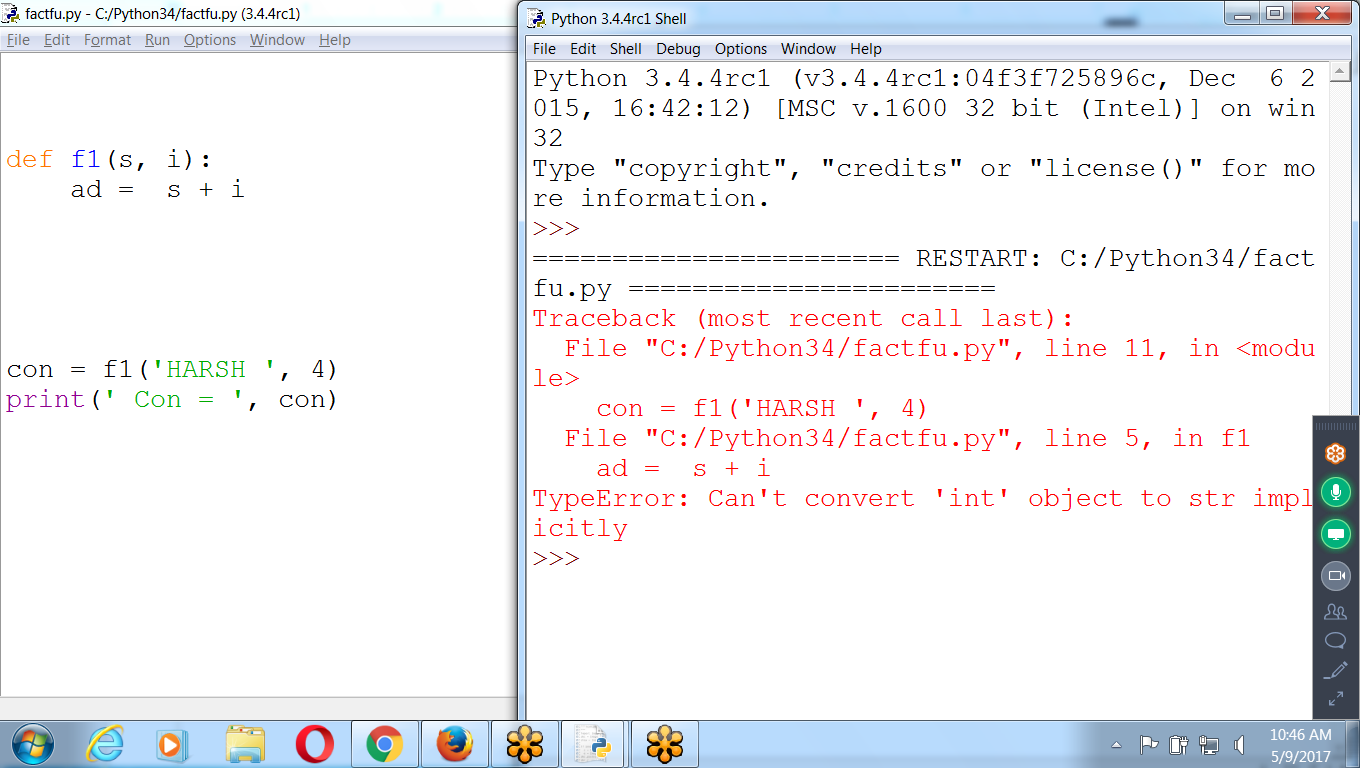
# Less number of Receiving Parameters



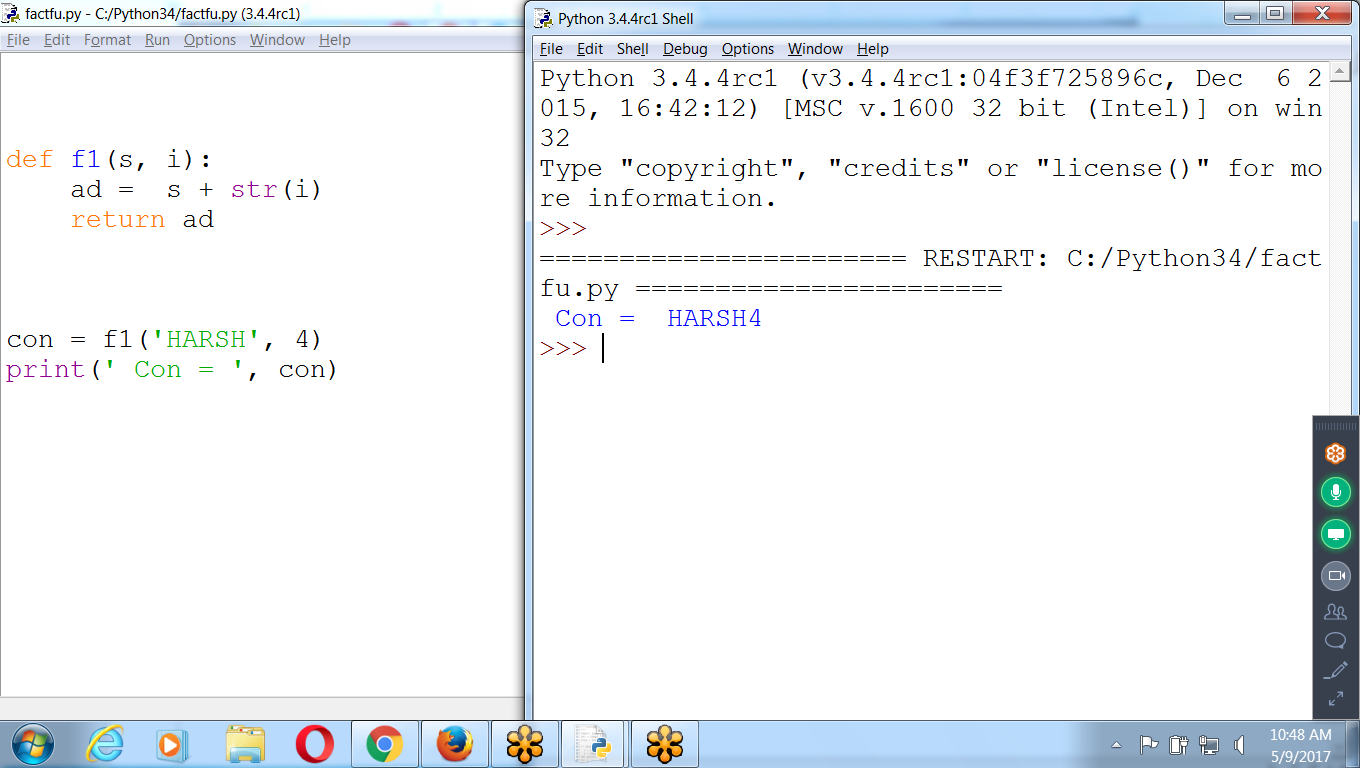
# Returning Less number of Parameters



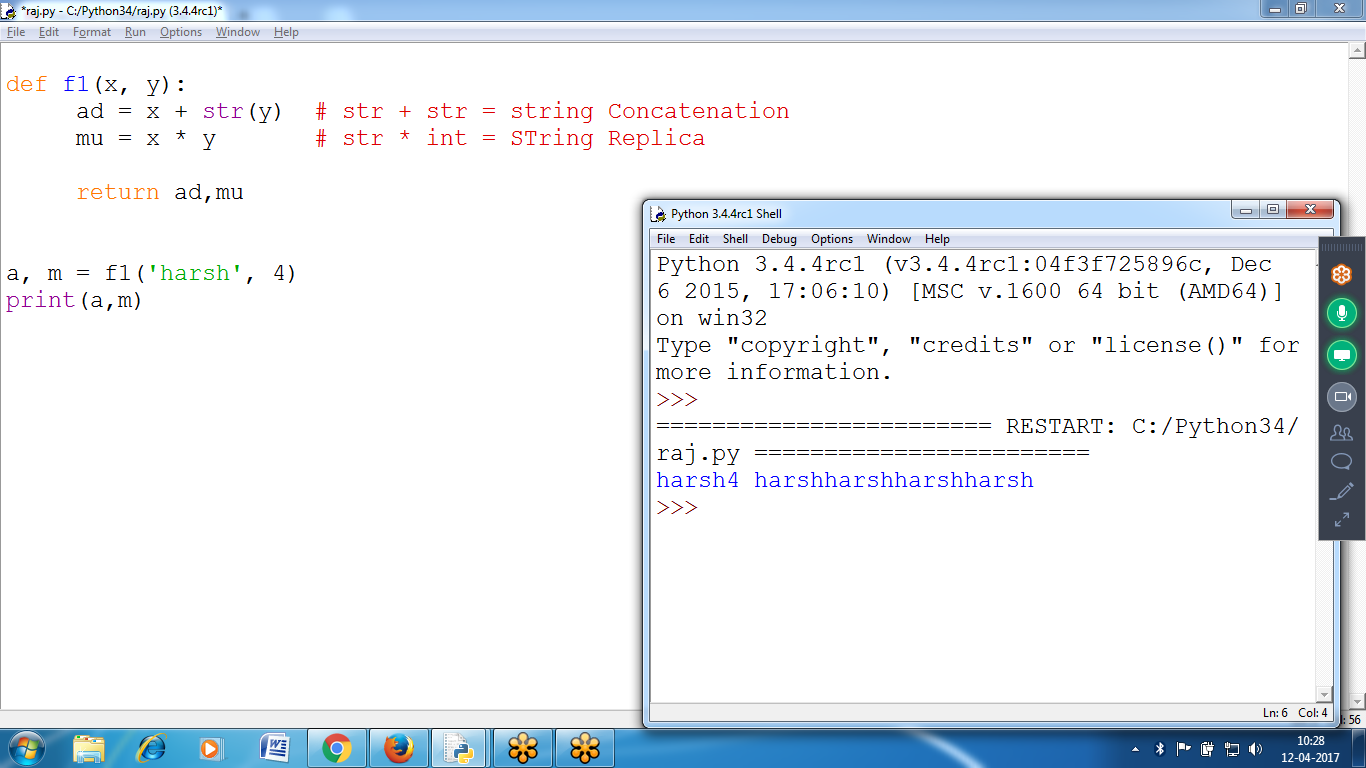
STring + integer :: Error



String + String = concatenation



Multiple Return Values String Concatenation and Replica



def f1(x, y):

ad = x + str(y) # str + str = string Concatenation

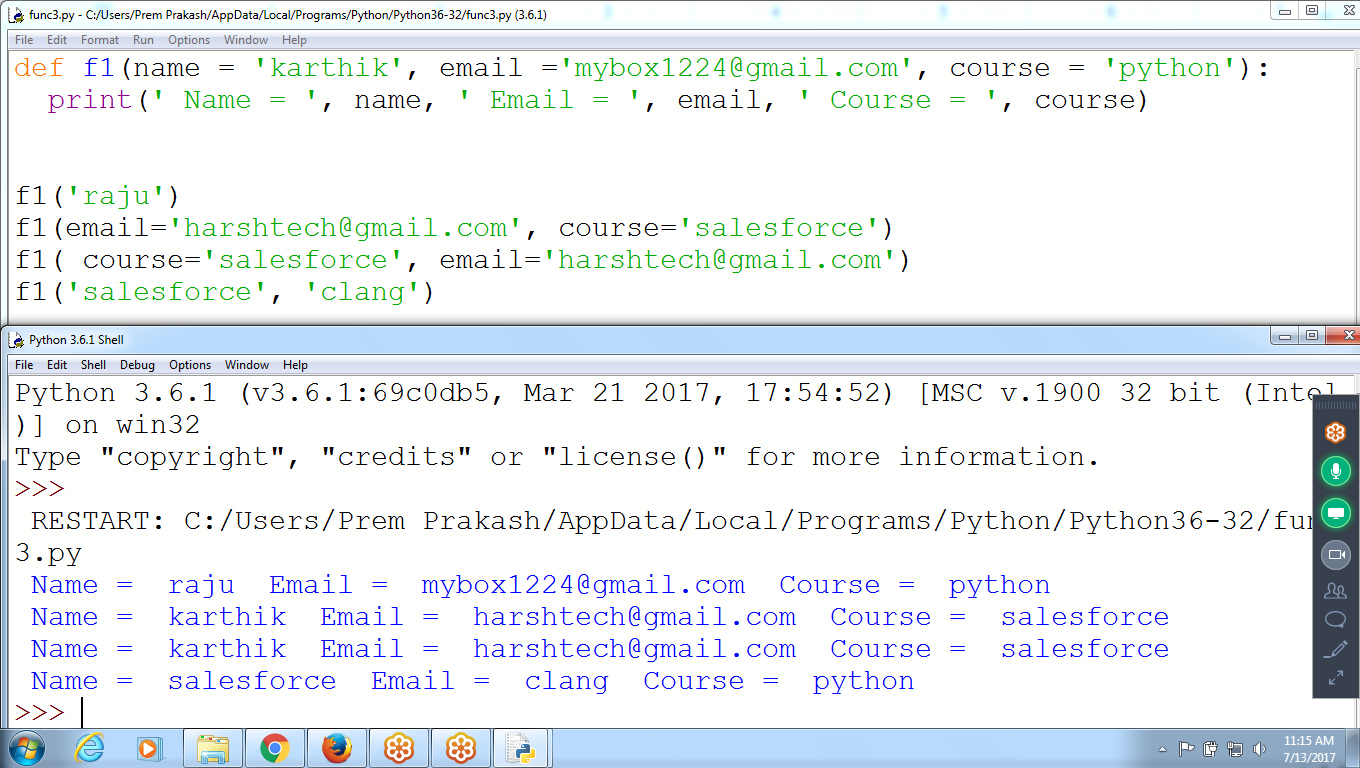
mu = x \* y # str \* int = STring Replica

return ad,mu

a, m = f1('harsh', 4)

print(a,m)

Keyword Arguments



def f1(name = 'karthik', email ='mybox1224@gmail.com', course = 'python'):

print(' Name = ', name, ' Email = ', email, ' Course = ', course)

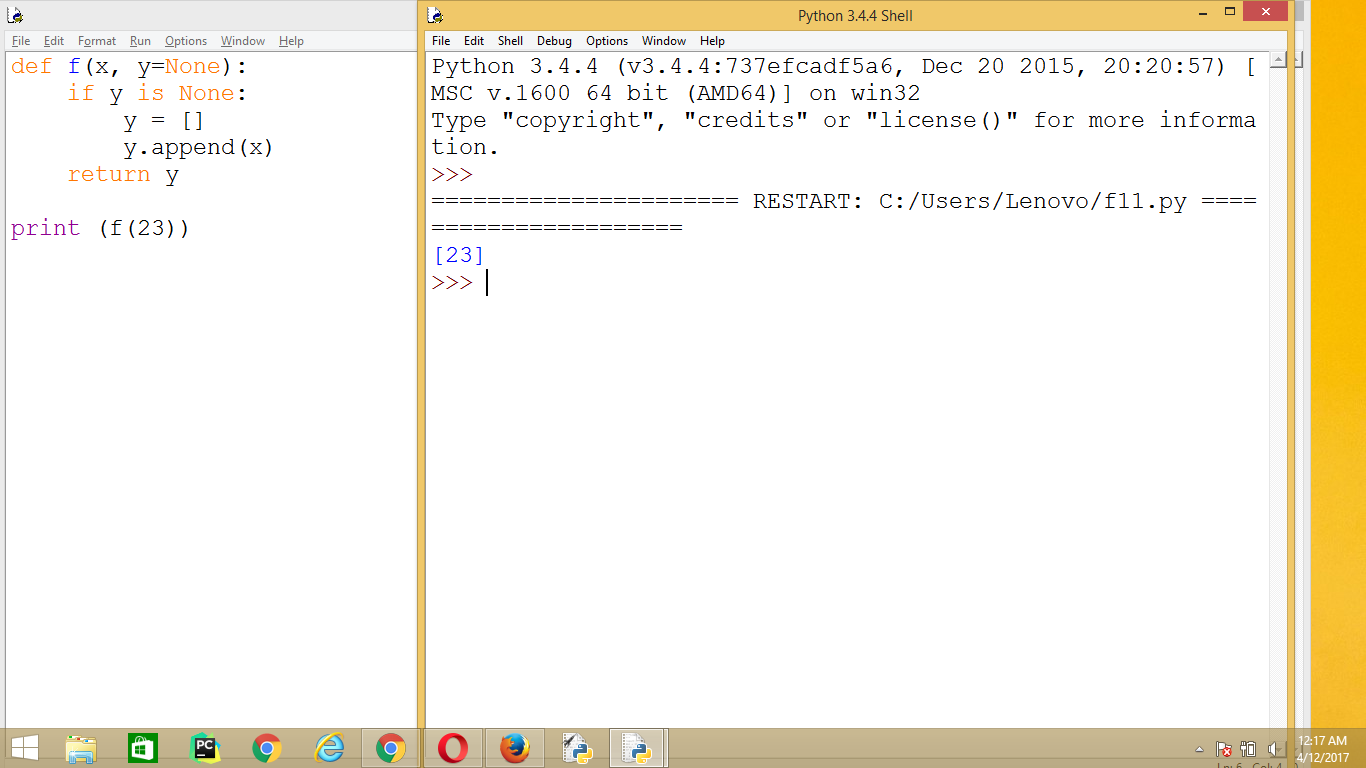
f1('raju')

f1(email='harshtech@gmail.com', course='salesforce')

f1( course='salesforce', email='harshtech@gmail.com')

f1('salesforce', 'clang')

Default Arguments



def f(x, y=None):

if y is None:

y = []

y.append(x)

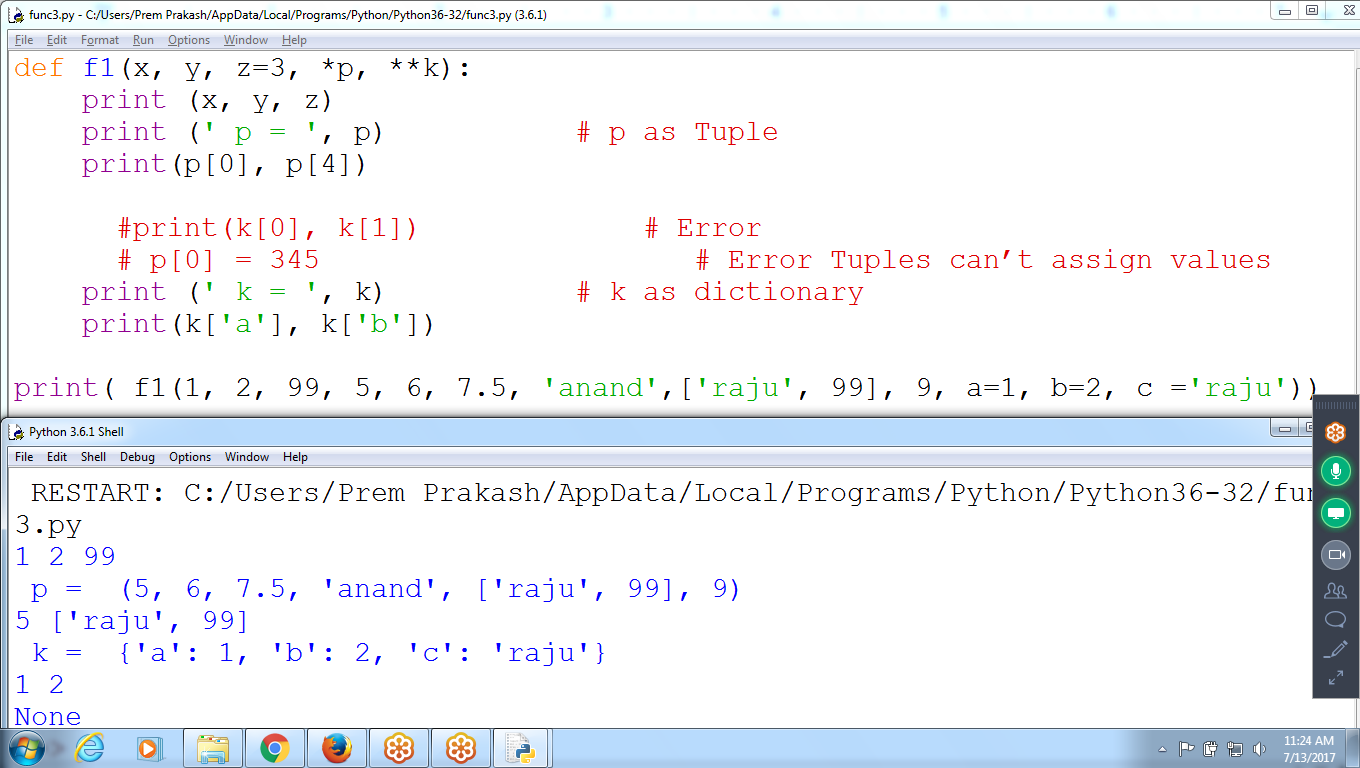
return y

print (f(23))

Mixed Parameter Passing List and Dictionary

\*P as TUPLE

\*\*k as Dictionary



def f1(x, y, z=3, \*p, \*\*k):

print (x, y, z)

print (' p = ', p) # p as Tuple

print(p[0], p[4])

#print(k[0], k[1]) # Error

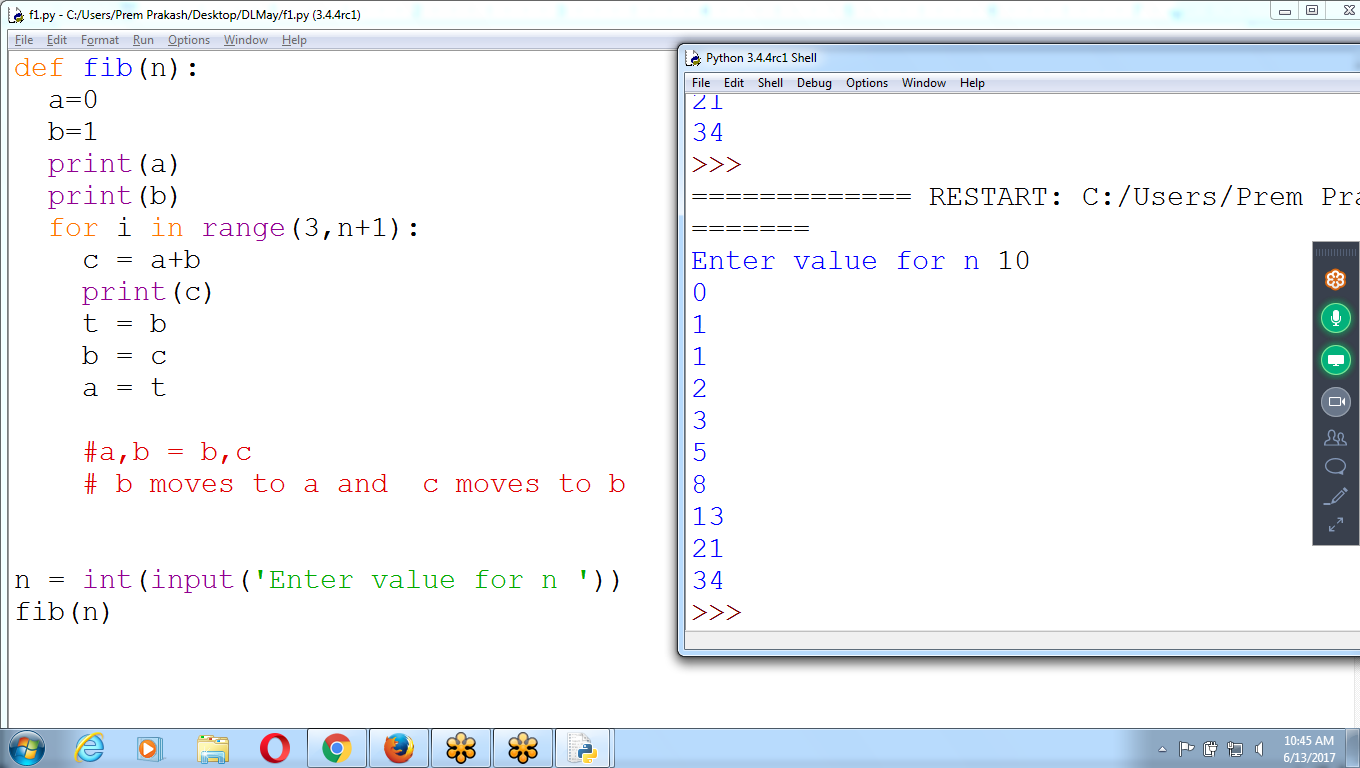
# p[0] = 345 # Error Tuples can’t assign values

print (' k = ', k) # k as dictionary

print(k['a'], k['b'])

print( f1(1, 2, 99, 5, 6, 7.5, 'anand',['raju', 99], 9, a=1, b=2, c ='raju'))

Fibonacci Series elements using Temporary Variable



def fib(n):

a=0

b=1

print(a)

print(b)

for i in range(3,n+1):

c = a+b

print(c)

t = b

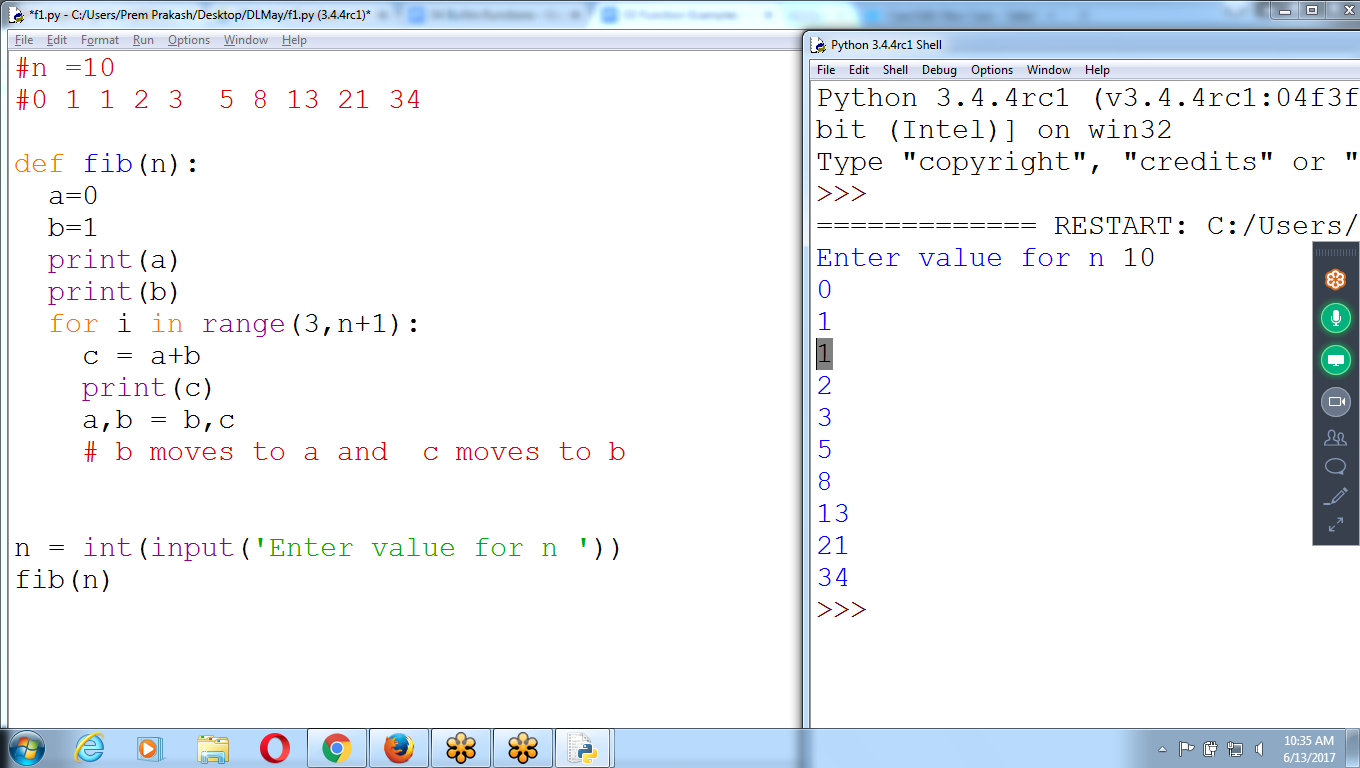
b = c

a = t

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

fib(n)

Without Temporary Variable



#n =10

#0 1 1 2 3 5 8 13 21 34

def fib(n):

a=0

b=1

print(a)

print(b)

for i in range(3,n+1):

c = a+b

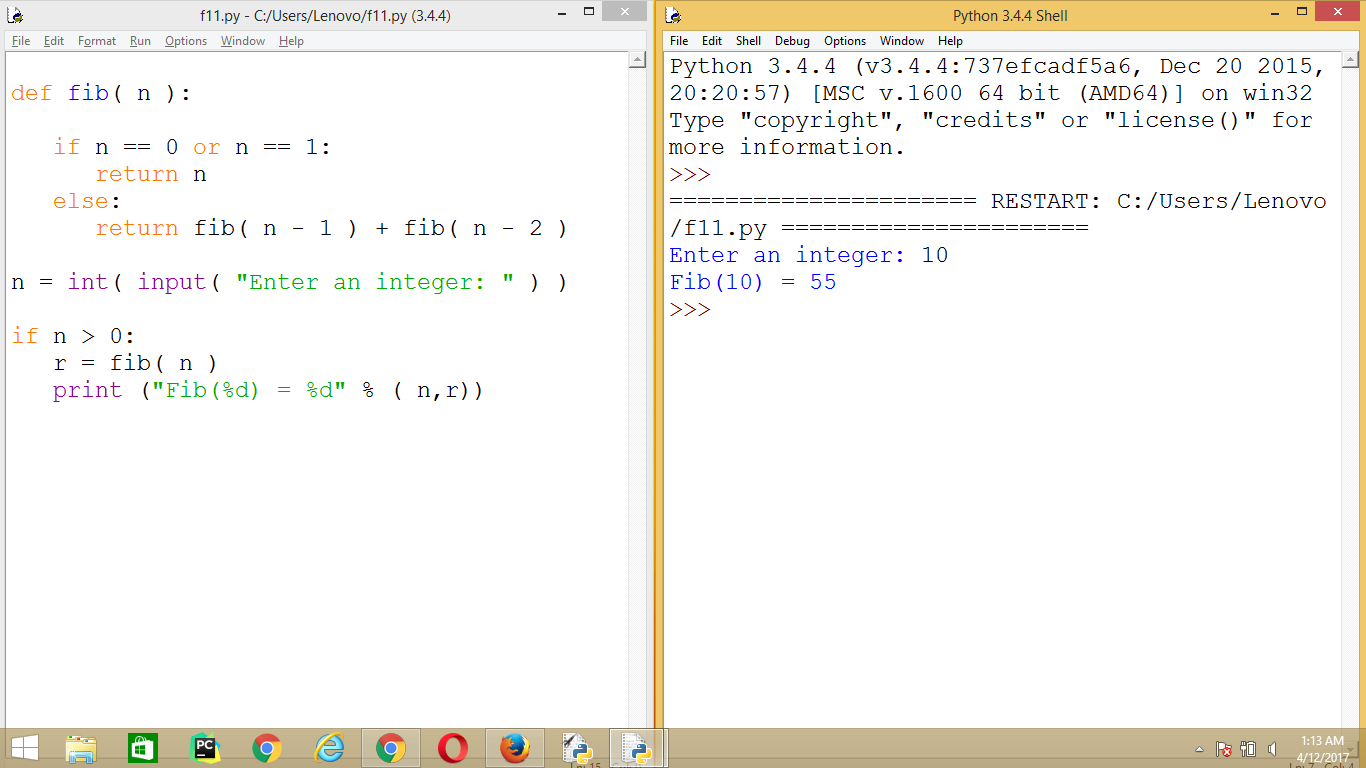
print(c)

a,b = b,c

# b moves to a and c moves to b

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

fib(n)



def fib( n ):

if n == 0 or n == 1:

return n

else:

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

n = int( input( "Enter an integer: " ) )

if n > 0:

r = fib( n )

print ("Fib(%d) = %d" % ( n,r))

A [function](http://www.programiz.com/python-programming/function) is a group of statements that perform a specific task. They can either be user-defined or already built into Python interpreter.

Functions that come built into the Python language itself are called built-in functions and are readily available to us.

Functions like print(), input(), eval() etc. that we have been using

|  |  |
| --- | --- |
| **Built-in Function** | **Description** |
| abs() | Return the absolute value of a number. |
| all() | Return True if all elements of the iterable are true (or if the iterable is empty). |
| any() | Return True if any element of the iterable is true. If the iterable is empty, return False. |
| ascii() | Return a string containing a printable representation of an object, but escape the non-ASCII characters. |
| bin() | Convert an integer number to a binary string. |
| bool() | Convert a value to a Boolean. |
| bytearray() | Return a new array of bytes. |
| bytes() | Return a new "bytes" object. |
| callable() | Return True if the object argument appears callable, False if not. |
| chr() | Return the string representing a character. |
| classmethod() | Return a class method for the function. |
| compile() | Compile the source into a code or AST object. |
| complex() | Create a complex number or convert a string or number to a complex number. |
| delattr() | Deletes the named attribute of an object. |
| dict() | Create a new dictionary. |
| dir() | Return the list of names in the current local scope. |
| divmod() | Return a pair of numbers consisting of quotient and remainder when using integer division. |
| enumerate() | Return an enumerate object. |
| eval() | The argument is parsed and evaluated as a Python expression. |
| exec() | Dynamic execution of Python code. |
| filter() | Construct an iterator from elements of iterable for which function returns true. |
| float() | Convert a string or a number to floating point. |
| format() | Convert a value to a "formatted" representation. |
| frozenset() | Return a new frozenset object. |
| getattr() | Return the value of the named attribute of an object. |
| globals() | Return a dictionary representing the current global symbol table. |
| hasattr() | Return True if the name is one of the object's attributes. |
| hash() | Return the hash value of the object. |
| help() | Invoke the built-in help system. |
| hex() | Convert an integer number to a hexadecimal string. |
| id() | Return the "identity" of an object. |
| input() | Reads a line from input, converts it to a string (stripping a trailing newline), and returns that. |
| int() | Convert a number or string to an integer. |
| isinstance() | Return True if the object argument is an instance. |
| issubclass() | Return True if class is a subclass. |
| iter() | Return an iterator object. |
| len() | Return the length (the number of items) of an object. |
| list() | Return a list. |
| locals() | Update and return a dictionary representing the current local symbol table. |
| map() | Return an iterator that applies function to every item of iterable, yielding the results. |
| max() | Return the largest item in an iterable. |
| memoryview() | Return a "memory view" object created from the given argument. |
| min() | Return the smallest item in an iterable. |
| next() | Retrieve the next item from the iterator. |
| object() | Return a new featureless object. |
| oct() | Convert an integer number to an octal string. |
| open() | Open file and return a corresponding file object. |
| ord() | Return an integer representing the Unicode. |
| pow() | Return power raised to a number. |
| print() | Print objects to the stream. |
| property() | Return a property attribute. |
| range() | Return an iterable sequence. |
| repr() | Return a string containing a printable representation of an object. |
| reversed() | Return a reverse iterator. |
| round() | Return the rounded floating point value. |
| set() | Return a new set object. |
| setattr() | Assigns the value to the attribute. |
| slice() | Return a slice object. |
| sorted() | Return a new sorted list. |
| staticmethod() | Return a static method for function. |
| str() | Return a str version of object. |
| sum() | Sums the items of an iterable from left to right and returns the total. |
| super() | Return a proxy object that delegates method calls to a parent or sibling class. |
| tuple() | Return a tuple |
| type() | Return the type of an object. |
| vars() | Return the \_\_dict\_\_ attribute for a module, class, instance, or any other object. |
| zip() | Make an iterator that aggregates elements from each of the iterables. |
| \_\_import\_\_() | This function is invoked by the import statement. |