# **Find Phone Number in a string**

To find a phone number in a string. the pattern: three numbers, a hyphen, three numbers, a hyphen, and four numbers. Here’s an example: 415-555-4242.

# CHeck given string having Phone Number or not

# 569Karthik 999-123-6789 and123-456

st = input('Enter any string ') #prem994-999-0032 anadn 970470

def fphone(r):

if(len(r) != 12):

return False

for i in range(0,3):

if(not r[i].isdecimal()):

return False

if(r[3] != '-'):

return False

for i in range(4,7):

if(not r[i].isdecimal()):

return False

if(r[7] != '-'):

return False

for i in range(8,12):

if(not r[i].isdecimal()):

return False

return True

for i in range(len(st)):

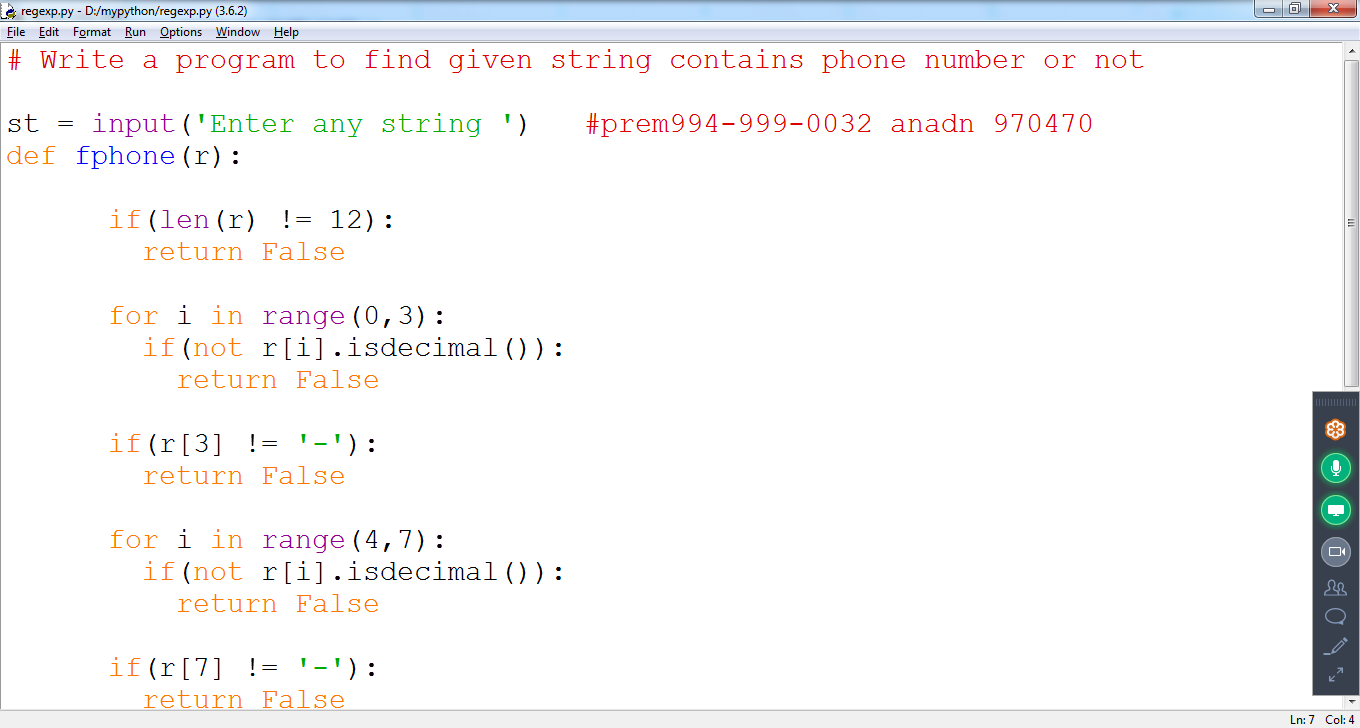
res = st[i:i+12]

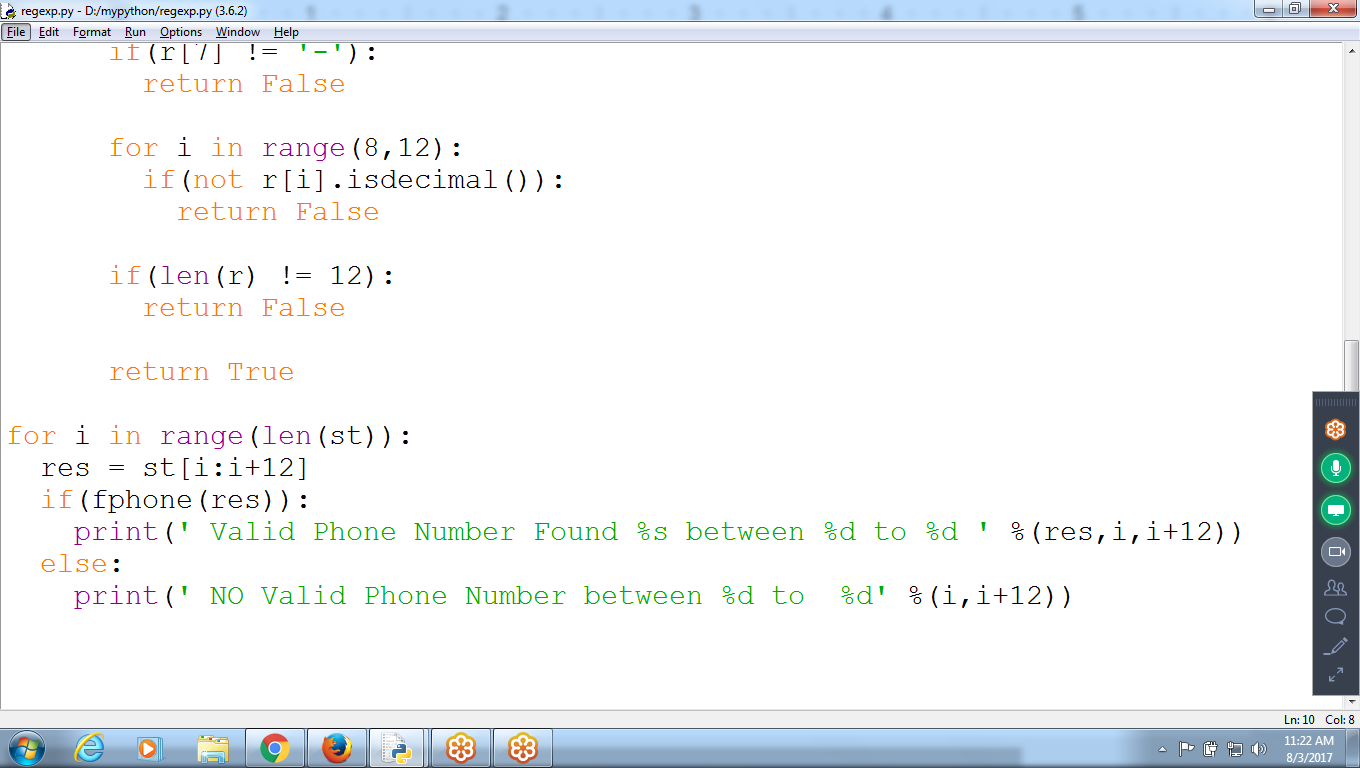
if(fphone(res)):

print(' Valid Phone Number Found %s between %d to %d ' %(res,i,i+12))

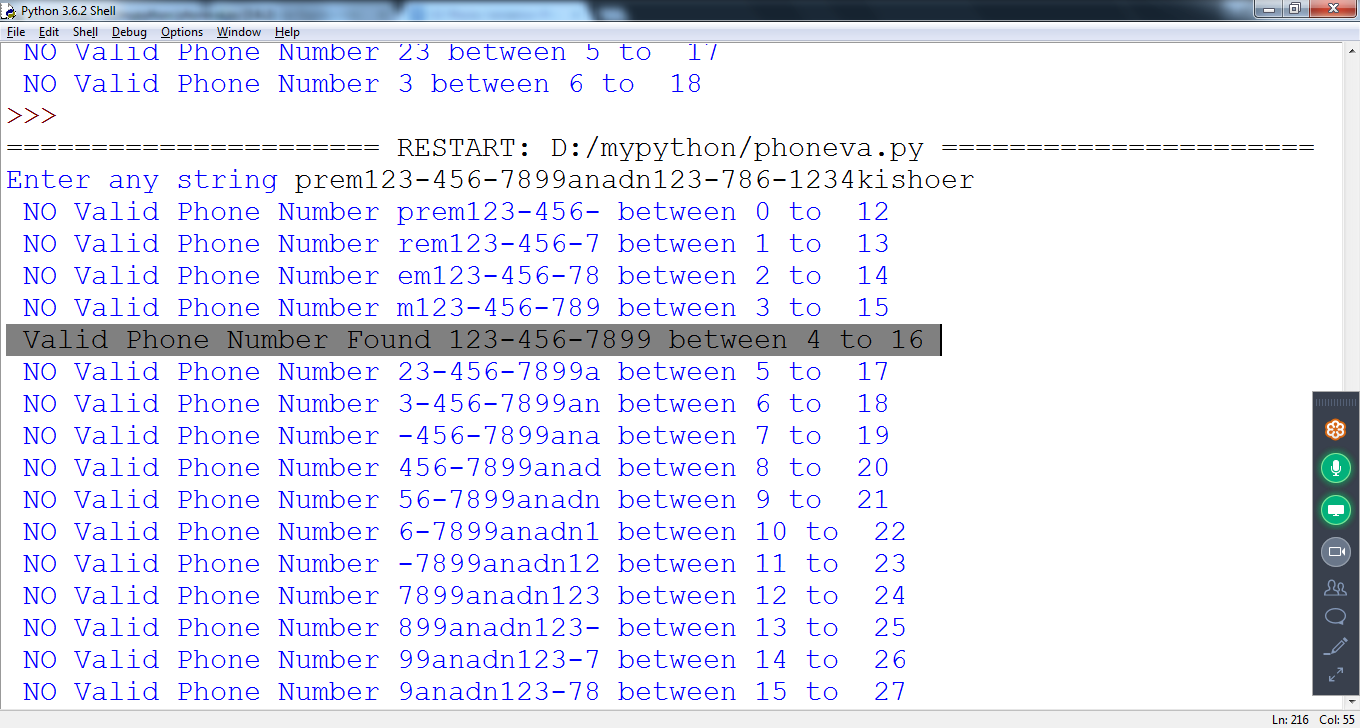
else:

print(' NO Valid Phone Number %s between %d to %d' %(res,i,i+12))



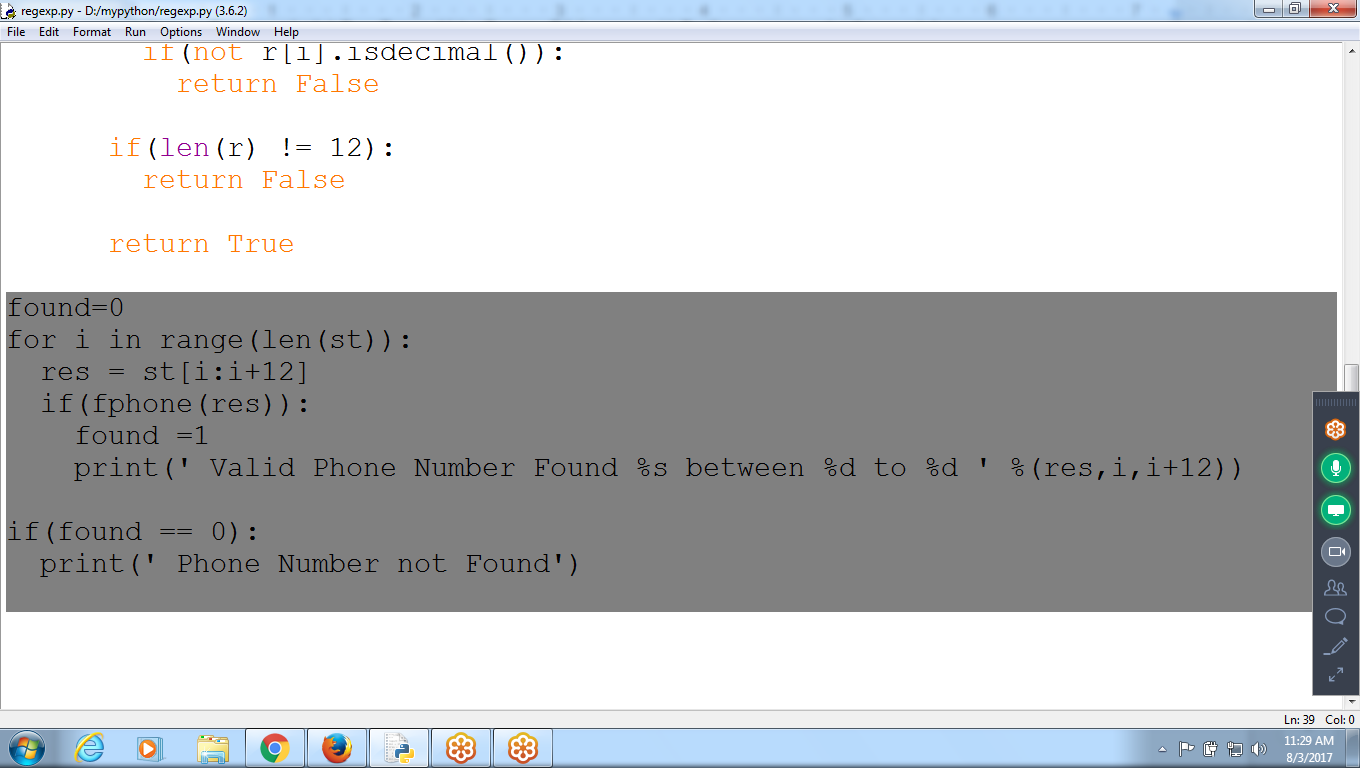


Result:

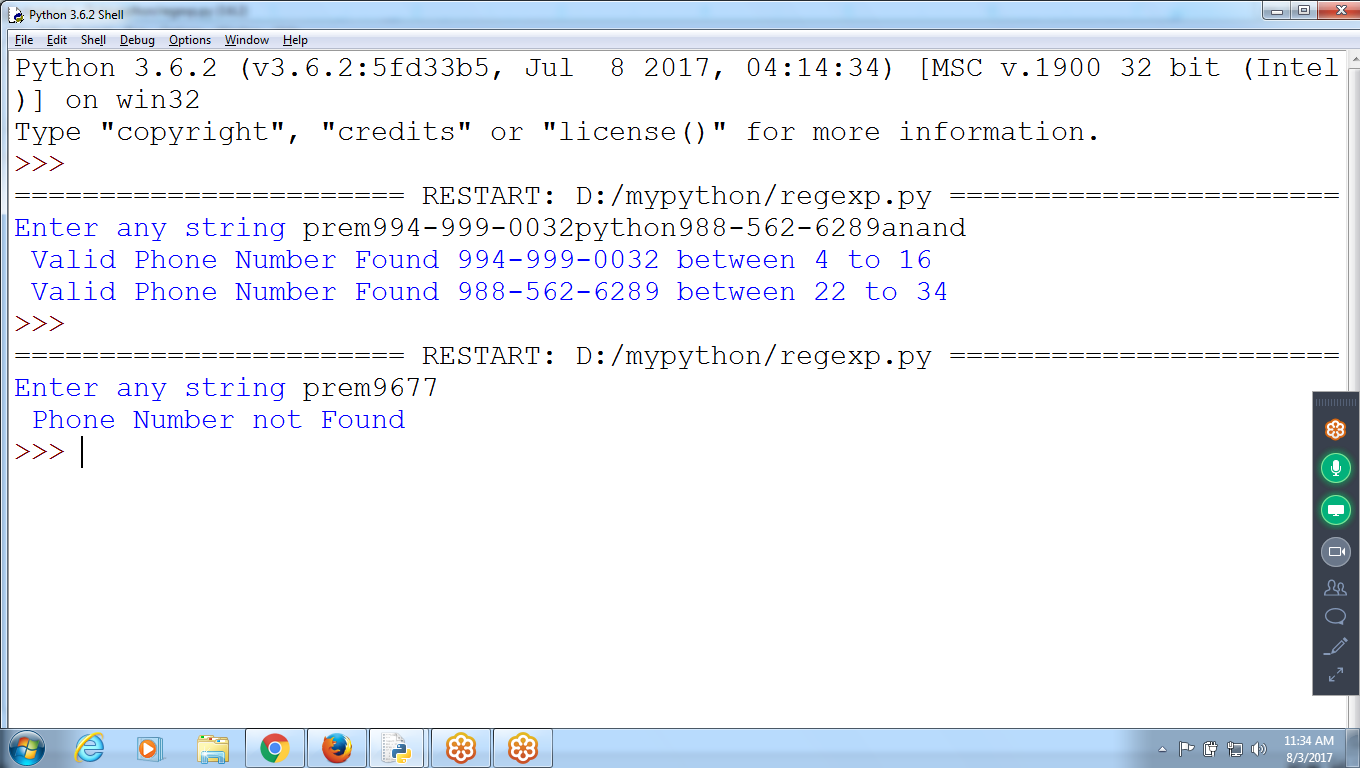


**Second Option::: should not display Phone number not found Every time**

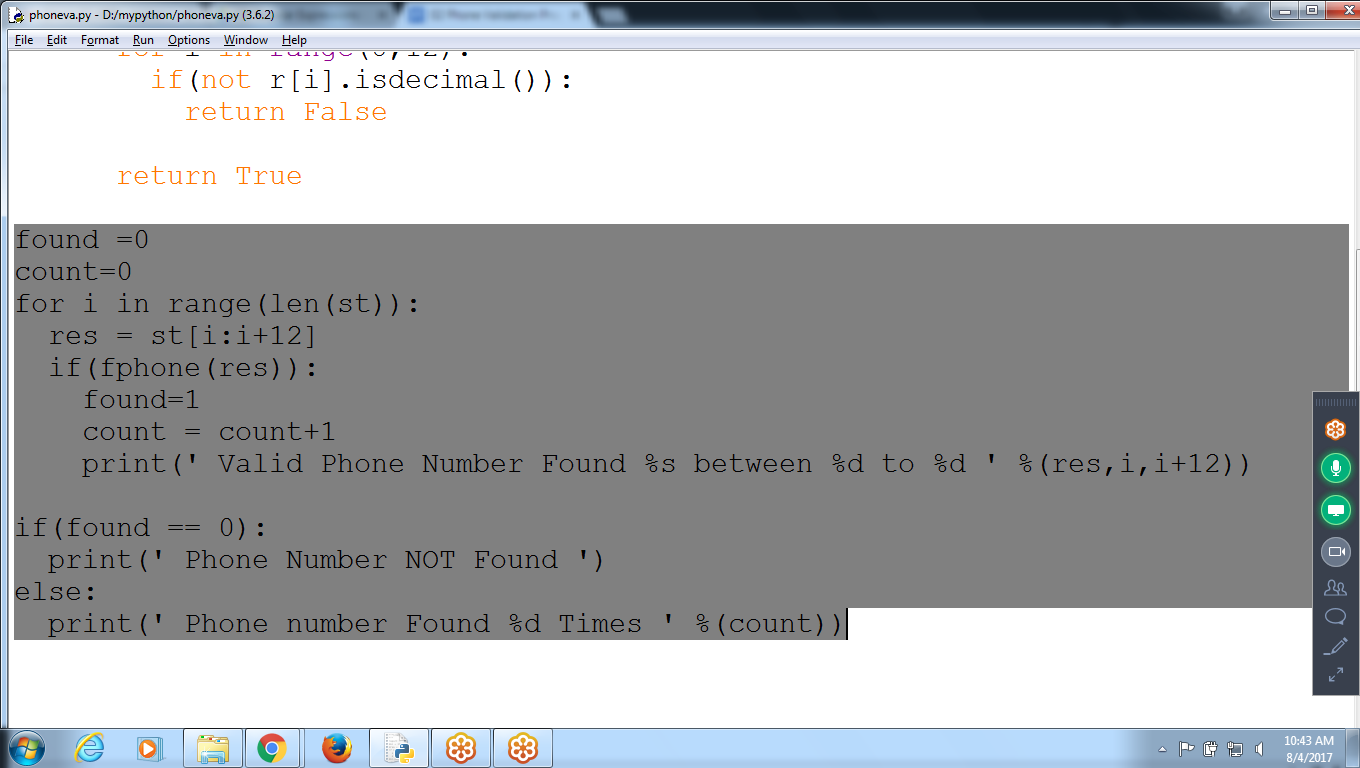
**Change The Code**

****

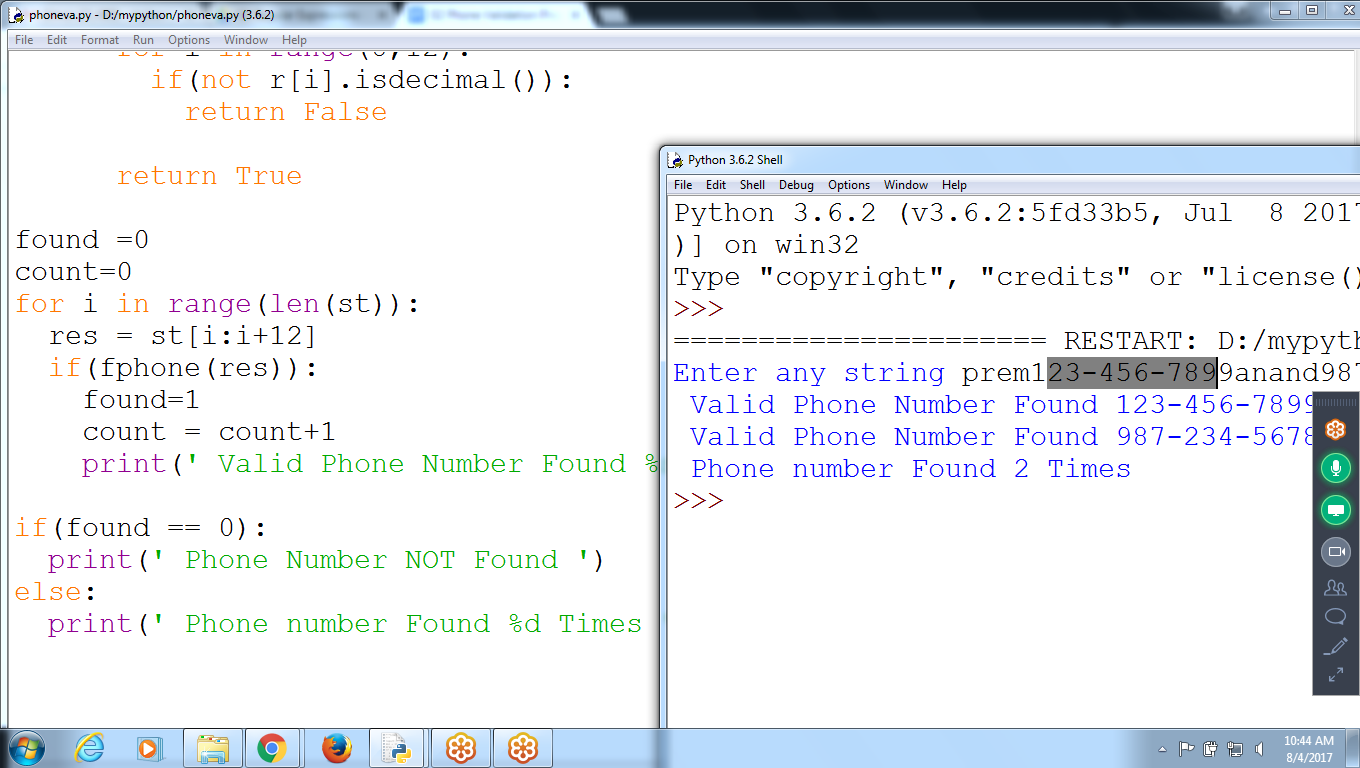
**Result::**

****

**THird: How many Times Phone Number exist**

****

**Result:**

****

**Return Messages from Fphone function()**

**# CHeck given string having Phone Number or not**

**# 569Karthik 999-123-6789 and123-456**

**st = input('Enter any string ') #prem994-999-0032 anadn 970470**

**def fphone(r):**

**if(len(r) != 12):**

**return 'Invalid Length of Characters '**

**for i in range(0,3):**

**if(not r[i].isdecimal()):**

**return 'Not Valid Digits between 0 to 3 '**

**if(r[3] != '-'):**

**return 'Hyphen NOt Found at 4'**

**for i in range(4,7):**

**if(not r[i].isdecimal()):**

**return 'Not Valid Digits between 5 to 7 '**

**if(r[7] != '-'):**

**return 'Hyphen NOt Found at 8 '**

**for i in range(8,12):**

**if(not r[i].isdecimal()):**

**return 'Not Valid Digits between 9 to 11 '**

**return 'Valid'**

**found =0**

**count=0**

**for i in range(len(st)):**

**res = st[i:i+12]**

**mes = fphone(res)**

**if(mes == 'Valid'):**

**found=1**

**count = count+1**

**print(' Message = ',mes)**

**print(' Valid Phone Number Found %s between %d to %d ' %(res,i,i+12))**

**else:**

**print(' Message = ',mes)**

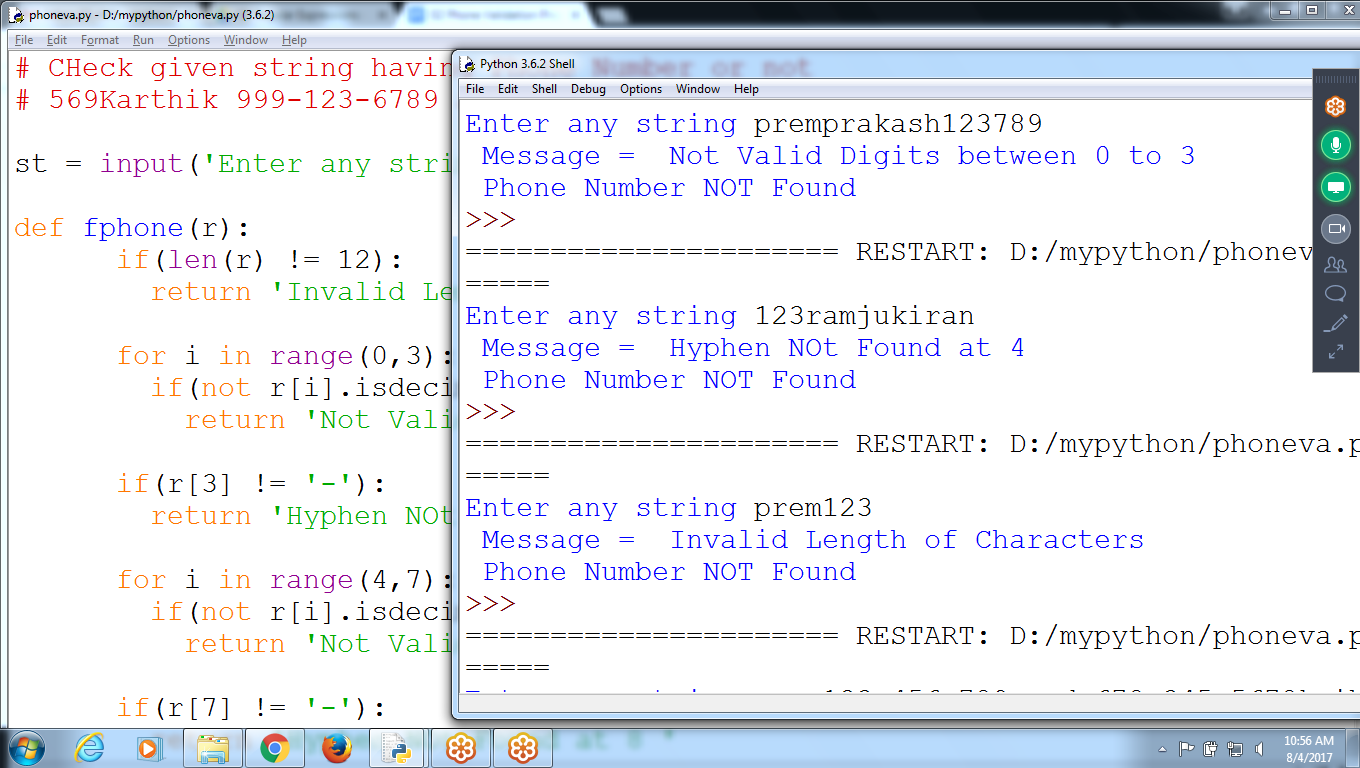
**break**

**if(found == 0):**

**print(' Phone Number NOT Found ')**

**else:**

**print(' Phone number Found %d Times ' %(count))**

****

# **Regular Expressions**

a phone number formatted like 415.555.4242 or (415) 555-4242? if the phone number had an extension, like 415-555-4242 x99? The phone() function **would fail** to validate them

* Regular expressions, called *regexes* for short, are descriptions for a pattern of text.
* For example, **\d** in a regex stands for a digit character—that is, any **single numeral 0 to 9**

**\d\d\d-\d\d\d-\d\d \d\d (or) \d{3}-\d{3}-\d{4}**

## **Creating Regex Objects**

* All the regex functions in Python are in the **re** module
* Passing a string value representing regular expression to **re.compile()** returns a Regex pattern object
* Search for Phone number in given string

>>> **phoneNumRegex = re.compile('\d\d\d-\d\d\d-\d\d\d\d')**

Now the phoneNumRegex variable contains a Regex object.

Typing r'\d\d\d-\d\d\d-\d\d\d\d' is much easier than typing '\\d\\d\\d-\\d\\d\\d-\\d\\d\\d\\d'

**#re is Regular Expression Library**

**# compile(), search()**

import re

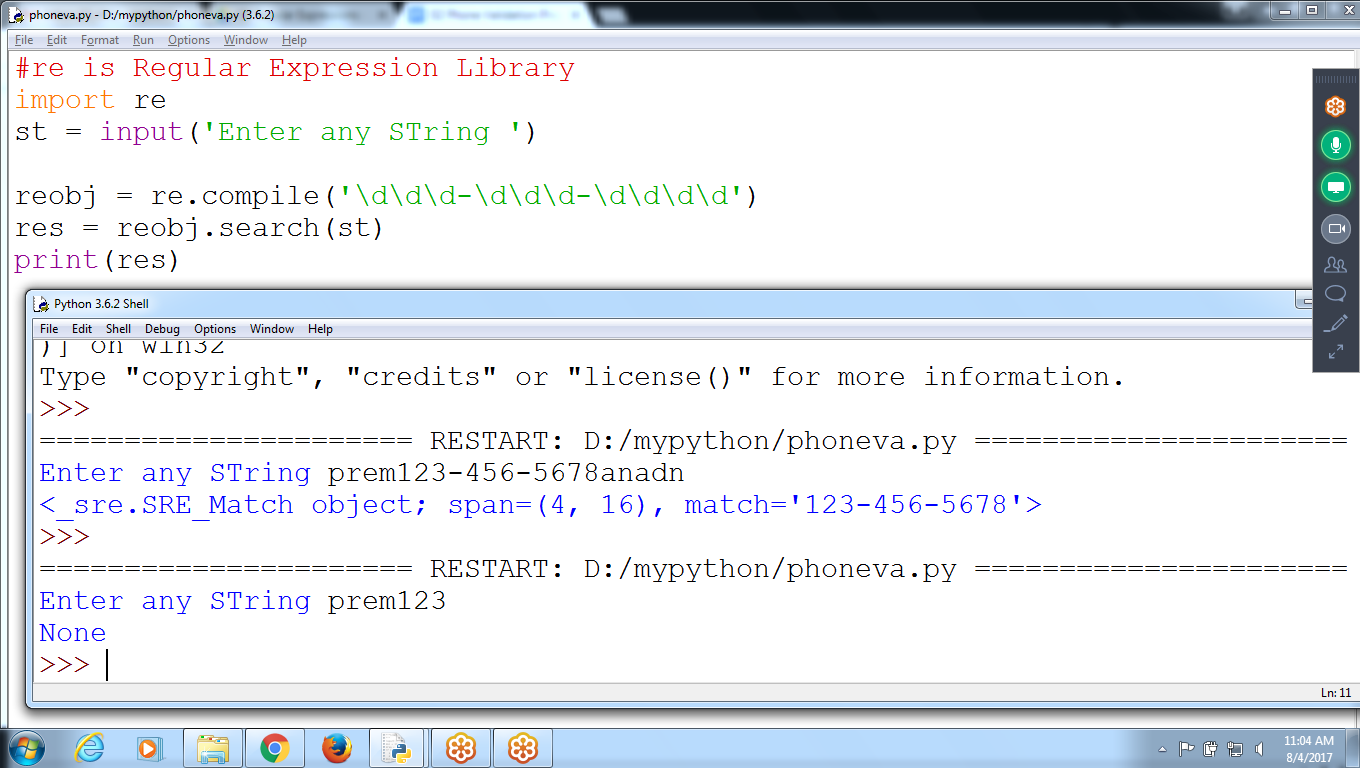
st = input('Enter any STring ')

reobj = re.compile('\d\d\d-\d\d\d-\d\d\d\d')

res = reobj.search(st)

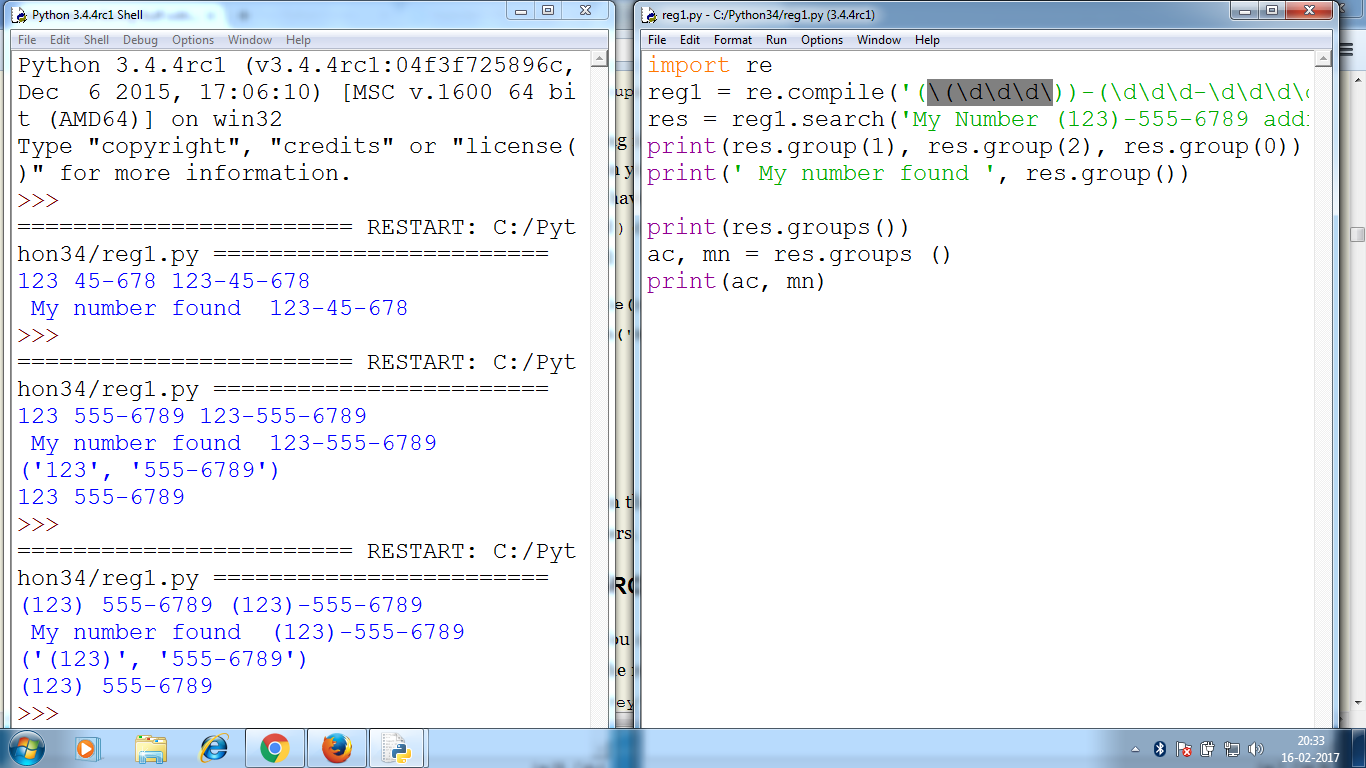
print(res)

**If Exists returns Position where it is existing else returns NONE**



**if you need to match a parenthesis in text?**

The \( and \) escape characters in the raw string passed to re.compile()

****

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

default\_order = "{}, {} and {}".format('Prem','Anand','Chaitu')

print('\n--- Default Order ---')

print(default\_order)

# order using positional argument

positional\_order = "{1}, {0} and {2}".format('Prem','Anand','Chaitu')

print('\n--- Positional Order ---')

print(positional\_order)

# order using keyword argument

keyword\_order = "{a}, {c} and {p}".format(p='Prem', a='Anand', c='Chaitu')

print('\n--- Keyword Order ---')

print(keyword\_order)

--- Default Order ---

Prem, Anand and Chaitu

--- Positional Order ---

Anand, Prem and Chaitu

--- Keyword Order ---

Anand, Chaitu and Prem

