## **The startswith() and endswith() String Methods**

The startswith() and endswith() methods return True if the string value they are called on begins or ends (respectively) with the string passed to the method

>>> st='Hello World'

>>> st.startswith('hello') # Lower case h

False

>>> st.startswith('Hello')

True

>>> st.endswith('World')

True

>>>

## 

## 

## **The join() and split() String Methods**

**String Concatenation**

>>> 'hello' + 'world' + 'python'

'helloworldpython'

* The join() method is useful when a list of strings that need to be joined together into a single string
* The join() method is called on a string, gets passed a list of strings, and returns a string

>>> ':::'.join('prem','prakash','python')

Traceback (most recent call last):

File "<pyshell#4>", line 1, in <module>

':::'.join('prem','prakash','python')

**TypeError: join() takes exactly one argument (3 given)**

>>>

**JOIN ::: LIST of Strings into Single String**

>>> ':::'.join**(['prem','prakash','python'])**

'prem:::prakash:::python'

>>>

>>> ':'.join(['prem','prakash','karhtik'])

'Prem:prakash:karhtik'

>>> '->'.join(['prem','prakash','karhtik'])

'prem->prakash->karhtik'

>>>

>>> " p\*3 ".join(['prem','prakash','python'])

'prem p\*3 prakash p\*3 python'

>>>

>>> ' '.join(['java', 'python', 'c#'])

'java python c#'

>>> 'pr'.join(['java', 'python', 'c#'])

'java**pr**pythonprc#'

>>>

Split called on a string value and returns a list of strings

Given string splits into list of strings

>>> 'prem p\*3 prakash p\*3 python'.split()  **#Default space**

['prem', 'p\*3', 'prakash', 'p\*3', 'python']

>>> 'prem p\*3 prakash p\*3 python'.split('p')

['', 'rem ', '\*3 ', 'rakash ', '\*3 ', 'ython']

>>>

## **Justifying Text with rjust(), ljust(), and center()**

**Just means MOVE**

* The rjust() and ljust() string methods return a padded version of the string they are called on, with **spaces inserted** to justify the text.
* The first argument to both methods is an **integer** length for the justified string

>>> 'python progam'.ljust(40) # Characters moves to Left side

'python progam '

>>> 'python progam'.center(40)

' python progam '

>>> 'python progam'.rjust(40)

' python progam'

>>>

>>> 'language'.rjust(25,'\*')

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*language'

>>> 'language'.ljust(25,'\*')

'language\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

>>>

>>> 'Hello'.**center(20**)  
' Hello '  
>>> 'Hello'.center(20, '=')  
'=======Hello========'

## **Removing Whitespace with strip(), rstrip(), and lstrip()**

To strip off whitespace characters (space, tab, and newline) from the left side, right side, or both sides of a string.

>>> str=' Hello World '

>>> str.**strip() # Removes space left and right**

'Hello World'

>>> str

' Hello World '

>>> str.**lstrip()**

'Hello World '

>>> str.**rstrip()**

' Hello World'

>>> str

' Hello World '

>>>

**Built-in String Methods in PYTHON**

Python includes the following built-in methods to manipulate strings −

1 capitalize()

Capitalizes first letter of string

2 center(width, fillchar)

Returns a space-padded string with the original string centered to a total of width columns.

3 count(str, beg= 0,end=len(string))

Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given.

4 decode(encoding='UTF-8',errors='strict')

Decodes the string using the codec registered for encoding. encoding defaults to the default string encoding.

5 encode(encoding='UTF-8',errors='strict')

Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'.

6 endswith(suffix, beg=0, end=len(string))

Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; returns true if so and false otherwise.

7 expandtabs(tabsize=8)

Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided.

8 **find(str, beg=0 end=len(string))**

Determine if str occurs in string or in a substring of string if starting index beg and ending index end are given returns index if found and -1 otherwise.

>>> fs = 'helloworld hello world '

>>> fs.find('world')

**5**

**>>>**

**String not existing returns ‘-1’**

>>> fs = 'helloworld hello world '

>>> fs.find('lol')

-1

9 index(str, beg=0, end=len(string))

Same as find(), but **raises an exception if** str not found.

>>> fs = 'helloworld hello world '

>>> fs.index('world')

5

>>>

>>> fs.index('lol')

Traceback (most recent call last):

File "<pyshell#65>", line 1, in <module>

fs.index('lol')

ValueError: substring not found

>>>

10 isalnum()

Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.

11 isalpha()

Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.

12 **isdigit()**

Returns true if string contains only digits and false otherwise.

>>> fs = '123'

>**>> fs.isdigit()**

True

>>>

>>> **fs.isdecimal()**

True

>>>

>>>

>>> **not(fs.isalpha())**

True

>>>

13 islower()

Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.

14 isnumeric()

Returns true if a unicode string contains only numeric characters and false otherwise.

15 isspace()

Returns true if string contains only whitespace characters and false otherwise.

16 istitle()

Returns true if string is properly "titlecased" and false otherwise.

17 isupper()

Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise.

18 join(seq)

Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string.

19 len(string)

Returns the length of the string

20 ljust(width[, fillchar])

Returns a space-padded string with the original string left-justified to a total of width columns.

2 1lower()

Converts all uppercase letters in string to lowercase.

22 lstrip()

Removes all leading whitespace in string.

**23 maketrans()**

Returns a translation table to be used in translate function.

from string import maketrans # Required to call maketrans function.

Fs = ‘helloworld’

intab = "aeiou"  
outtab = "12345"  
trantab = fs.maketrans(intab, outtab)  
  
str = "this is string example....wow!!!"  
print str.translate(trantab)

Result…..th3s 3s str3ng 2x1mpl2....w4w!!!

24 max(str)

Returns the max alphabetical character from the string str.

25 min(str)

Returns the min alphabetical character from the string str.

>>> max('hello world')

'w'

>>>

>>>

>>> min('helloworld')

'd'

>>>

26 replace(old, new [, max])

Replaces all occurrences of old in string with new or at most max occurrences if max given.

>>> fs = 'hello world hello world '

>>> fs.replace('o', 'z')

'hellz wzrld hellz wzrld '

>>>

27 rfind(str, beg=0,end=len(string))

Same as find(), but search backwards in string.

>>> fs = 'hello world hello world '

>>> fs.find('world')

6

>>>

>>>

>>> fs.rfind('world')

18

>>>

28 rindex( str, beg=0, end=len(string))

Same as index(), but search backwards in string.

29 rjust(width,[, fillchar])

Returns a space-padded string with the original string right-justified to a total of width columns.

30 rstrip()

Removes all trailing whitespace of string.

31 split(str="", num=string.count(str))

Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given.

32 splitlines( num=string.count('\n'))

Splits string at all (or num) NEWLINEs and returns a list of each line with NEWLINEs removed.

33 startswith(str, beg=0,end=len(string))

Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise.

34 strip([chars])

Performs both lstrip() and rstrip() on string

35 swapcase()

Inverts case for all letters in string.

>>> fs = 'HellO WoRld'

>>> fs.swapcase()

'hELLo wOrLD'

>>>

36 title()

Returns "titlecased" version of string, that is, all words begin with uppercase and the rest are lowercase.

37 translate(table, deletechars="")

Translates string according to translation table str(256 chars), removing those in the del string.

38 upper()

Converts lowercase letters in string to uppercase.

39 zfill (width)

Returns original string leftpadded with zeros to a total of width characters; intended for numbers, zfill() retains any sign given (less one zero).

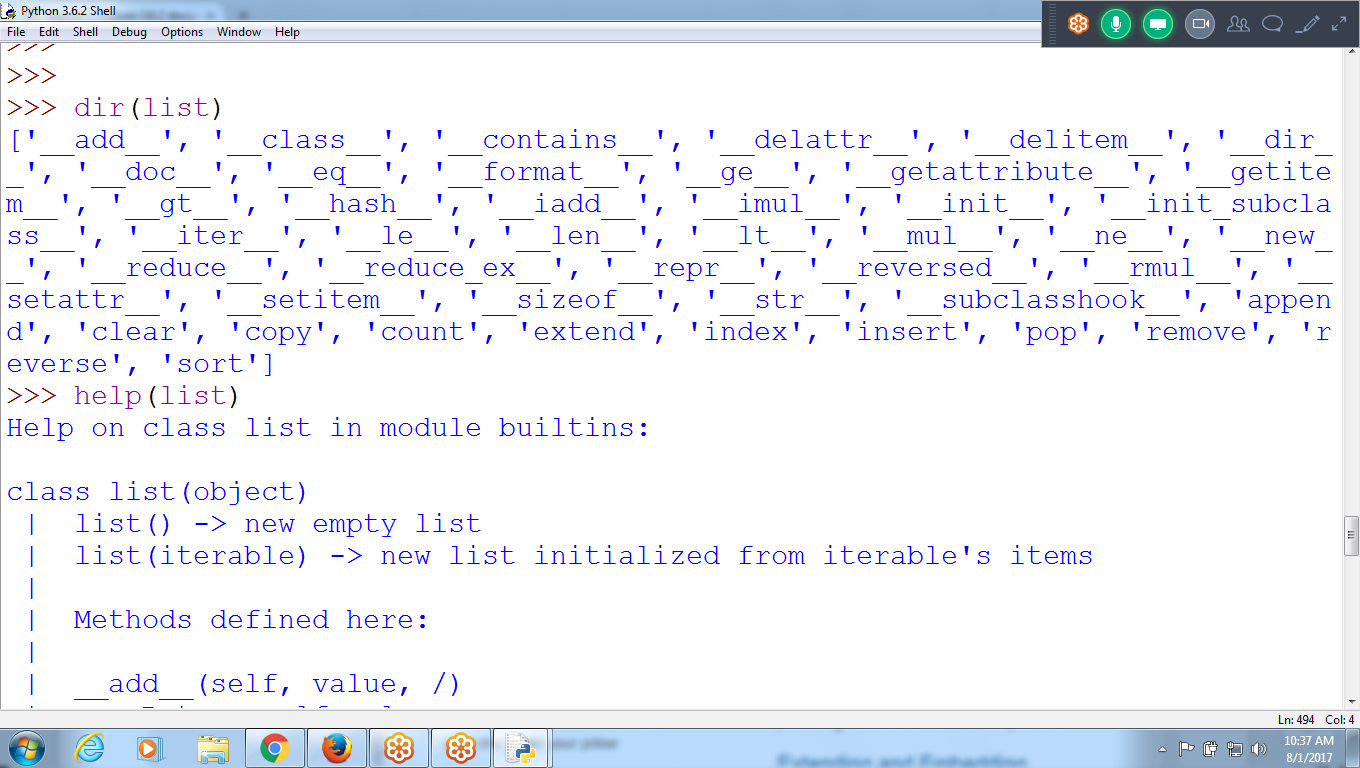
40 isdecimal()

Returns true if a unicode string contains only decimal characters and false otherwise.a

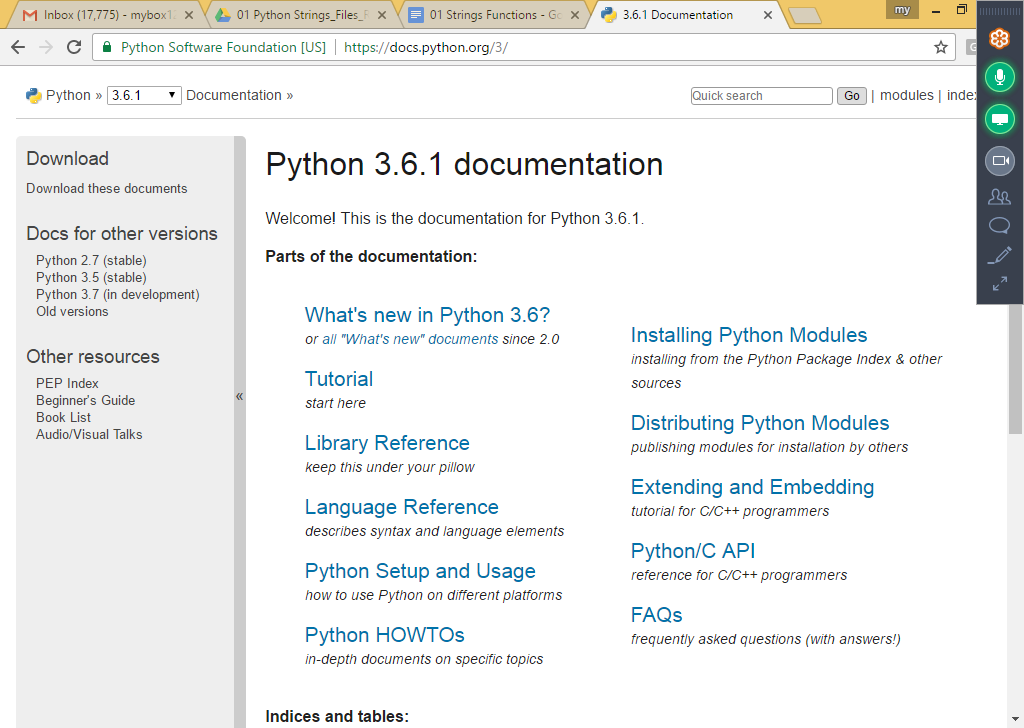
dir(list)

(or)

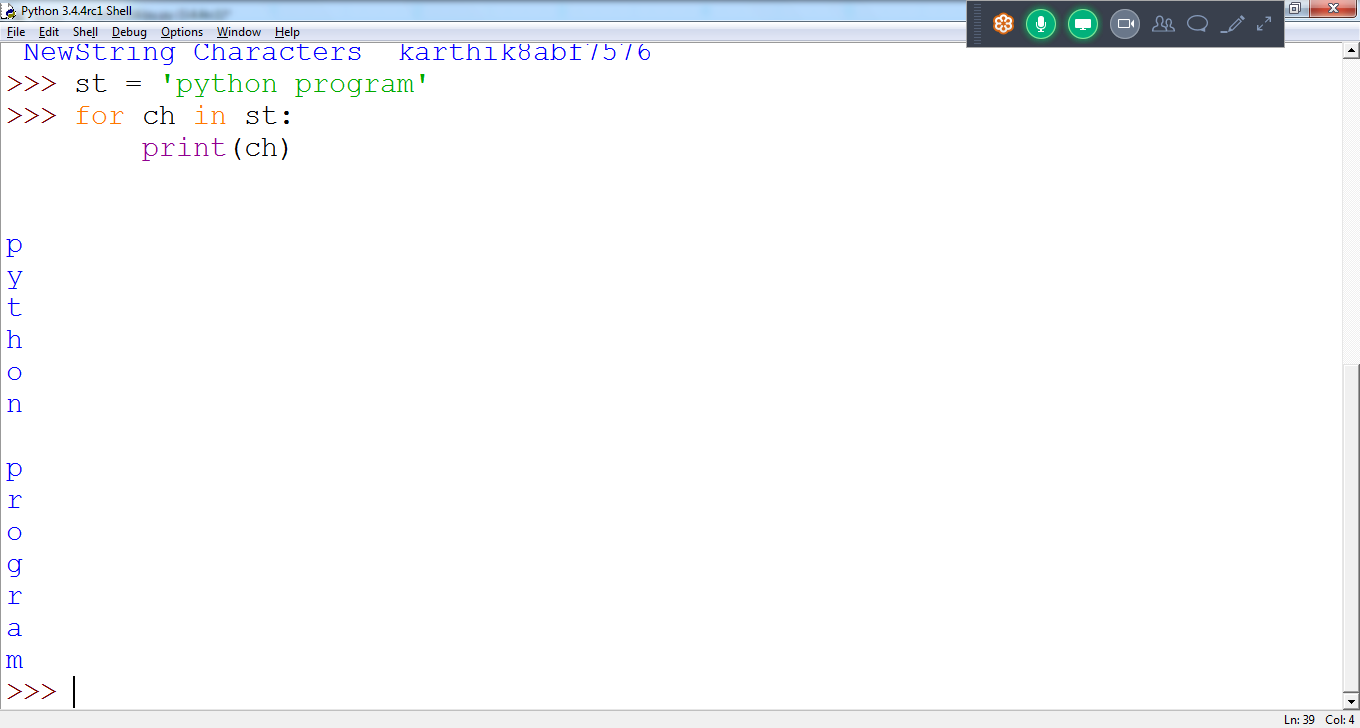
help(list)



<https://docs.python.org/3/>



**For loop with String and LIST**

****

**>>> for a in fs:**

**print(a + ' Good ')**

**h Good**

**e Good**

**l Good**

**l Good**

**o Good**

**Good**

**w Good**

**o Good**

**r Good**

**l Good**

**d Good**

**List :: Ls is collection of Strings**

**>>> ls = ['hello', 'good morning', 'python lang']**

**>>>**

**>>> for l in ls:**

**print(l)**

**hello**

**good morning**

**python lang**

**>>>**

**>>>**

**>>> for l in ls:**

**for ch in l:**

**print(ch)**

**h**

**e**

**l**

**l**

**o**

**g**

**o**

**>>> for l in ls:**

**print(l)**

**raju**

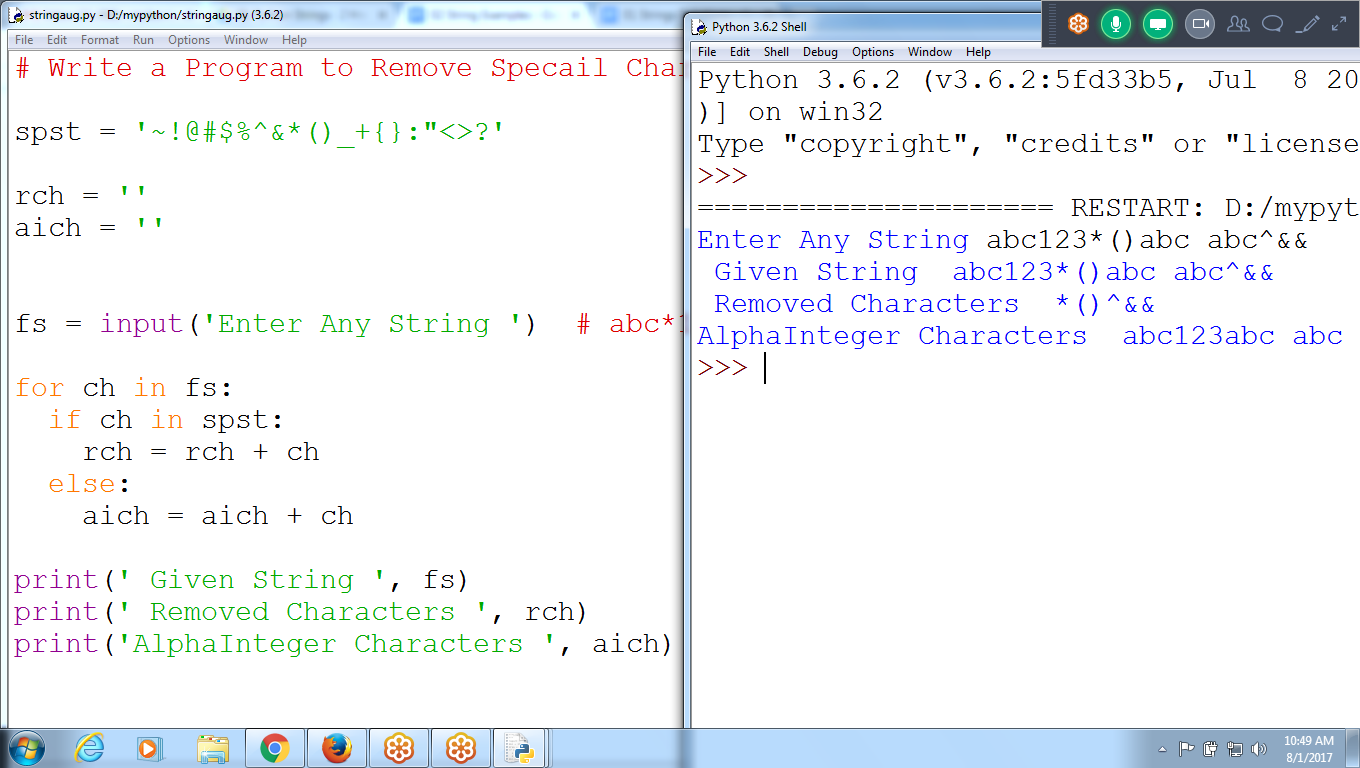
**rani**

**1234**

**jeevan**

**>>>**

* **python Program to Remove Special Characters From a String**

****

**# Remove Special Characters in a Given string**

st = input('Enter any string ') #\*prem12&\*(&\*^\*&\*&

newst = '' #prem

remsp = ''

spch = '~!@#$%^&\*()\_+123'

for ch **in** st:

if( ch **not in** spch):

newst = newst + ch # String Concatenation

else:

remsp = remsp + ch

print(' st = ',st)

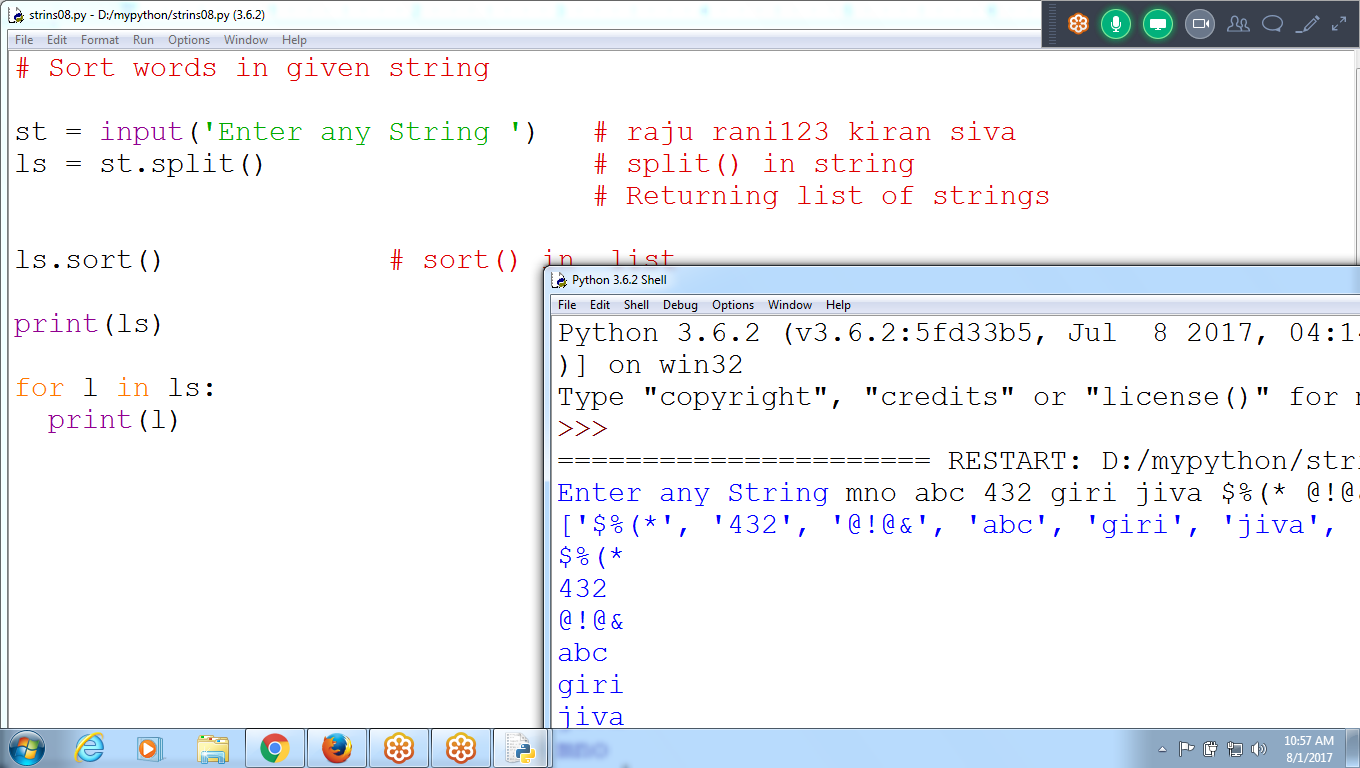
print(' Removed Characters = ', remsp)

print(' Without Spch = ', newst)

**Program to Sort Words in Alphabetic Order**

**split() returning collection of string, place in LIST type**

**sort() is method in LIST**

****

**# Sort words in given string**

**st = input('Enter any String ') # raju rani123 kiran siva**

**ls = st.split() # split() in string**

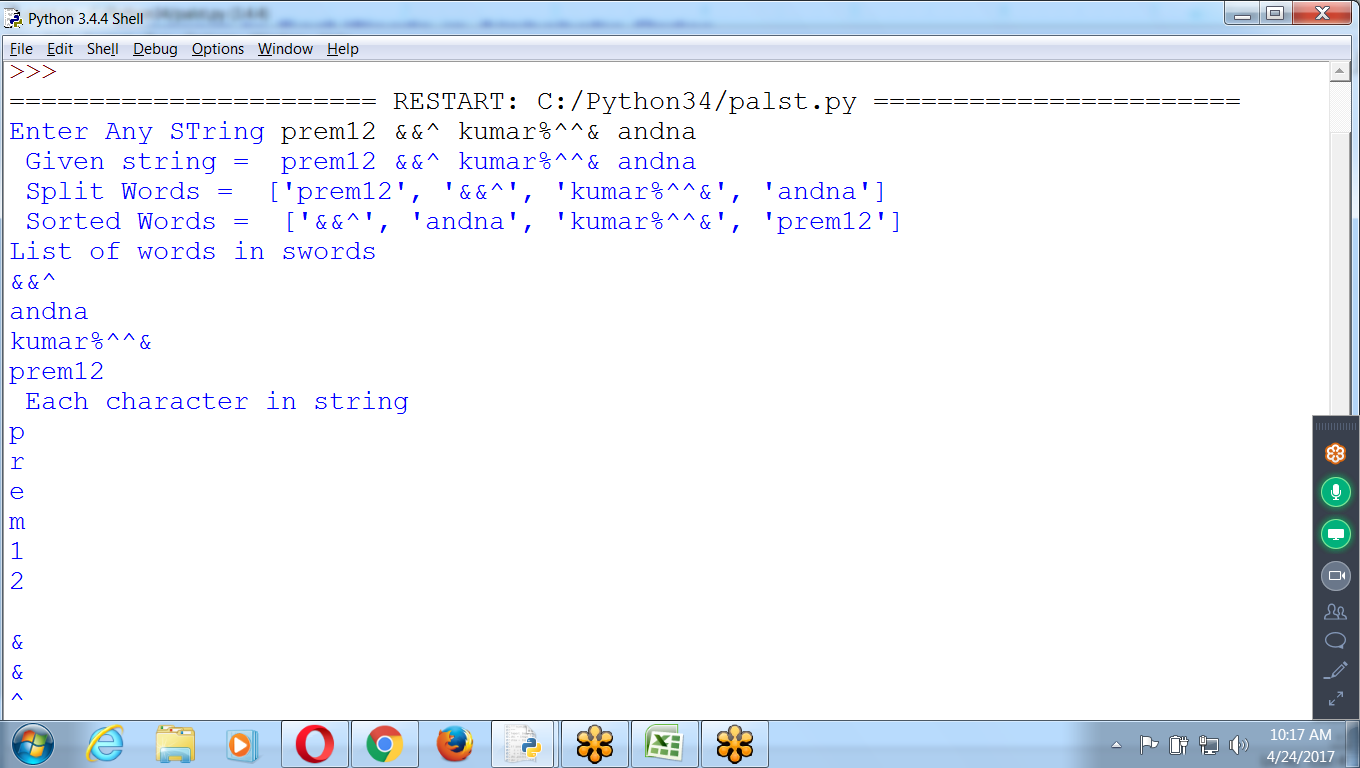
**# Returning list of strings**

**ls.sort() # sort() in list**

**print(ls)**

**for l in ls:**

**print(l)**



**# Input string and print words in SORTING Order**

**st = input('Enter Any STring ') # st is string**

**swords = st.split() # split() is method in string**

**print(' Given string = ', st)**

**print(' Split Words = ',swords) #swords is LIST**

**swords.sort() # sort() is method in List**

**print(' Sorted Words = ', swords)**

**print('List of words in swords ')**

**for w in swords: # w is is string**

**print(w)**

**print(' Each character in string ')**

**for ch in st: # ch is an character**

**print(ch)**

# 

M1 : Python : 30

M2 : Advance Python : 30

M3 : django : 30 H

M4: Projects in Python : 30 H

9.30 to 11 : python

11.30 to 1 : django

# 

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