String split methods

1. split()
2. rsplit()

split method required separator for splitting string into multiple strings.

Example:

```
>>> str1="a,b,c,d,e"
>>> list1=str1.split(",")
>>> print(list1)
['a', 'b', 'c', 'd', 'e']
>>> list2=str1.split(",",2)
>>> print(list2)
['a', 'b', 'c,d,e']
>>> list3=str1.rsplit(",")
>>> print(list3)
['a', 'b', 'c', 'd', 'e']
>>> list4=str1.rsplit(",",2)
>>> print(list4)
['a,b,c', 'd', 'e']
```

Example:

```
# example of input mutliple values in single line

str1=input("Enter values ")
print(str1)
list1=str1.split()
print(list1)
list2=list(map(int,list1))
print(list2)
```

Output

```
Enter values 10 20 30 40 50
10 20 30 40 50
['10', '20', '30', '40', '50']
[10, 20, 30, 40, 50]
```

Example:

```
# Write a program to input string and find pal words
# input: hello madam liril

str1=input("Enter any String ")
list1=str1.split()
for word in list1:
    if word==word[::-1]:
        print(word)
```

Output

```
Enter any String hello madam liril
madam
liril
```

join() method

This method is used to join list of strings (OR) collection of string into one string using separator.

Syntax: separator.join(iterable)

Separator is string which is used to join iterable of string into one string.

Example:

```
>>> list1=["hello","world"]
>>> str1=' '.join(list1)
>>> print(list1)
['hello', 'world']
>>> print(str1)
hello world
>>> str2=",".join(list1)
>>> print(str2)
hello,world
>>> list2=["a","b","c","d","e"]
>>> str3=",".join(list1)
>>> print(str3)
```

```
hello,world
>>> str3=",".join(list2)
>>> print(str3)
a,b,c,d,e
```

Example:

```
# Python program to print even length words in a string

str1=input("Enter any String ")
list1=str1.split()
for word in list1:
    if len(word)%2==0:
        print(word)
```

Output

```
Enter any String this is programming language
this
is
language
```

Example:

```
# Python Program to Accept the Strings Which Contains all Vowels

str1=input("Enter any String ")
str2=str1.lower()
if "a" in str2 and "e" in str2 and "i" in str2 and "o" in str2 and "u" in str2:
    print("Accepted...")
else:
    print("Not Accepted...")
```

Output

```
Enter any String aBcdEiOLU
Accepted...

Enter any String python
Not Accepted...
```

Example:

```
# Count the Number of matching characters in a pair of string
'''
Input : str1 = 'abcdef'
        str2 = 'defghia'
Output : 4
(i.e. matching characters :- a, d, e, f)
'''

str1=input("Enter first string ")
str2=input("Enter second string ")

# Removing duplicates
str3=''
for ch in str1:
    if ch not in str3:
        str3=str3+ch

# Removing duplicates
str4=''
for ch in str2:
    if ch not in str4:
        str4=str4+ch

# Reading character from string1 and searching in string2
c=0
for ch in str3:
    if ch in str4:
        c=c+1

print(c)
```

String alignment methods or justification methods

1. ljust()
2. rjust()
3. center()

str.ljust(*width*[, *fillchar*])
Return the string left justified in a string of length *width*. Padding is done using the specified *fillchar* (default is an ASCII space). The original string is returned if *width* is less than or equal to len(s).

Example:

```
s1="NIT"
print(s1.ljust(20))
print(s1.ljust(20,"*"))
print(s1.ljust(30,'$'))
print(s1.ljust(2))

names_list=["naresh","ramesh","ram","kishore","kiran"]
for name in names_list:
    print(name.ljust(20,'$'))
```

Output

```
NIT
NIT*****************
NIT$$$$$$$$$$$$$$$$$$$$$$$$$$
NIT
naresh$$$$$$$$$$$$$
ramesh$$$$$$$$$$$$$
ram$$$$$$$$$$$$$$$$
kishore$$$$$$$$$$$$
kiran$$$$$$$$$$$$$$
```

str.rjust(*width*[, *fillchar*])
Return the string right justified in a string of length *width*. Padding is done using the specified *fillchar* (default is an ASCII space). The original string is returned if *width* is less than or equal to len(s).

```
s1="NIT"
print(s1.rjust(20))
print(s1.rjust(20,'$'))
print(s1.rjust(20,'&'))
```

```
names_list=["naresh","ramesh","ram","kishore","kiran"]
for name in names_list:
    print(name.rjust(20,'$'))
```

Output

```
                NIT
$$$$$$$$$$$$$$$$NIT
&&&&&&&&&&&&&&&&&&NIT
$$$$$$$$$$$$naresh
$$$$$$$$$$$$ramesh
$$$$$$$$$$$$$$$ram
$$$$$$$$$$$kishore
$$$$$$$$$$$$$kiran
```

str.center(*width*[, *fillchar*])

Return centered in a string of length *width*. Padding is done using the specified *fillchar* (default is an ASCII space). The original string is returned if *width* is less than or equal to len(s).

Example:

```
str1="ABCD"
print(str1.center(20))
print(str1.center(20,'*'))
```

Output

```
        ABCD
********ABCD********
```

Strip methods or trim methods

These methods are used to remove leading or trailing characters from string.

1. lstrip()
2. rstrip()
3. strip()