strip methods

str.lstrip([*chars*])
Return a copy of the string with leading characters removed. The *chars* argument is a string specifying the set of characters to be removed. If omitted or None, the *chars* argument defaults to removing whitespace. The *chars* argument is not a prefix; rather, all combinations of its values are stripped:

```
>>> str1="     nit"
>>> str2="nit"
>>> str1==str2
False
>>> str1.lstrip()==str2
True
>>> str3="*****nit"
>>> str4="nit"
>>> str3==str4
False
>>> str3.lstrip("*")==str4
True
>>> str5="&*$&#$&nit"
>>> str6=str5.lstrip("$&#*")
>>> print(str5)
&*$&#$&nit
>>> print(str6)
nit
>>>
```

str.rstrip([*chars*])
Return a copy of the string with trailing characters removed. The *chars* argument is a string specifying the set of characters to be removed. If omitted or None, the *chars* argument defaults to removing whitespace. The *chars* argument is not a suffix; rather, all combinations of its values are stripped:

```
>>> s1="nit       "
>>> s2="nit"
>>> s1==s2
False
```

```
>>> s1.rstrip()==s2
True
>>> s3="nit****"
>>> s4=s3.rstrip("*")
>>> print(s3)
nit****
>>> print(s4)
nit
>>> s5="nit$#%@%$#&"
>>> s6=s5.rstrip("$#%@&")
>>> print(s5)
nit$#%@%$#&
>>> print(s6)
nit
>>> s7="nit$%%%*"
>>> s8=s7.rstrip("%")
>>> print(s7)
nit$%%%*
>>> print(s8)
nit$%%%*
>>>
```

str.strip([*chars*])
Return a copy of the string with the leading and trailing characters removed. The *chars* argument is a string specifying the set of characters to be removed. If omitted or None, the *chars* argument defaults to removing whitespace. The *chars* argument is not a prefix or suffix; rather, all combinations of its values are stripped

```
>>> str1="    nit     "
>>> str2="nit"
>>> str1==str2
False
>>> str1.strip()==str2
True
>>> s1="www.nareshit.com"
>>> s2=s1.strip("wc.om")
>>> print(s1)
www.nareshit.com
>>> print(s2)
```

```
nareshit
>>>
```

str.maketrans(*x*[, *y*[, *z*]])
This static method returns a translation table usable for str.translate().

str.translate(*table*)
Return a copy of the string in which each character has been mapped through the given translation table

```
>>> t1=str.maketrans("aeiou","@#$*%")
>>> t2=str.maketrans("@#$*%","aeiou")
>>> str1="programming"
>>> str2=str1.translate(t1)
>>> print(str1)
programming
>>> print(str2)
pr*gr@mm$ng
>>> str3=str2.translate(t2)
>>> print(str3)
programming
>>>
```

Example:

```
t1=str.maketrans("abcdefghijklmnopqrstuvwxyz","~!@#$%^&*()_+{}|?></[]\;'
.")
t2=str.maketrans("~!@#$%^&*()_+{}|?></[]\;'.","abcdefghijklmnopqrstuvwxy
z")
str1="python"
str2=str1.translate(t1)
print(str1,str2,sep="\n")
str3=str2.translate(t2)
print(str3)
```

Output:

```
python
|'/&}{
python
>>>
```

Conversion methods

str.upper()
Return a copy of the string with all the cased characters converted to uppercase

str.title()
Return a titlecased version of the string where words start with an uppercase character and the remaining characters are lowercase.

str.swapcase()
Return a copy of the string with uppercase characters converted to lowercase and vice versa.

str.lower()
Return a copy of the string with all the cased characters converted to lowercase

str.capitalize()
Return a copy of the string with its first character capitalized and the rest lowercased.

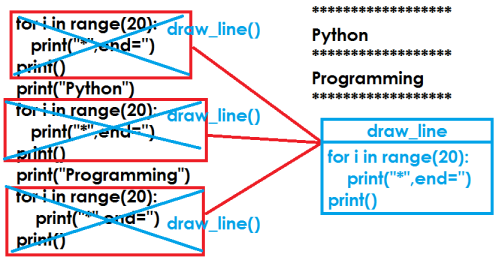
```
>>> str1="python"
>>> str2=str1.upper()
>>> print(str1,str2)
python PYTHON
>>> str3=str2.lower()
>>> print(str2,str3)
PYTHON python
>>> str4="python language"
>>> str5=str4.title()
>>> print(str4,str5)
python language Python Language
>>> str6="python language"
>>> str7=str6.capitalize()
>>> print(str6,str7)
python language Python language
>>> str8="pYTHON"
>>> str9=str8.swapcase()
>>> print(str8,str9)
```

pYTHON Python
>>>

Functions
Python multi paradigm programming language
Python support functional/procedural oriented programming
Functions are building blocks of procedural oriented programming/functional oriented programming

What is function?
A function is small program within program
A function is self contained block which contain set of instructions
A function is named block, which contains set of instructions used to perform operation

Advantage of functions
Reusability: Functions allows to write code one time and use number of times



Modularity: Dividing programming instructions according to their operations into small programs
Simplicity : Easy to understand
Efficient: Functions increase efficiency of the program by decrease size.

Types of functions
There are two types of functions
1. Predefined functions
2. User defined functions

Predefined functions are existing functions or the functions provided by python.
Eg: print(),input(),max(),sum(),min(),len(),….
The functions developed by programmer are called user defined functions

Syntax of defining function

```
def function-name([arguments]):
        '''doc string'''
        statement-1
        statement-2
```