**Methods of string class**

**str.capitalize()**

Return a copy of the string with its first character capitalized and the rest lowercased.

**Example:**

```
names_list=['naresh','SURESH','Kishore','ramesh']
for name in names_list:
    print(name.capitalize())
```

**Output:**

```
Naresh
Suresh
Kishore
Ramesh
>>>
```

**str.casefold()**

Return a casefolded copy of the string. Casefolded strings may be used for caseless matching.

```
>>> "nit".casefold()=="NIT".casefold()
True
>>>
>>> "NIT".casefold()
'nit'
>>> "nit".casefold()
'nit'
>>>
```

**Alignment methods**

**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).

**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).

**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).

**Example:**

```
student_dict={
    101:['naresh','python'],
    102:['suresh','java'],
    103:['kishore','c'],
    104:['ramesh','cpp']}
for rno,s in student_dict.items():
    print(str(rno).ljust(10),s[0].center(20),s[1].rjust(15))

for rno,s in student_dict.items():
    print(str(rno).ljust(10,'*'),s[0].center(20),s[1].rjust(15,'*'))
```

**Output:**

```
101              naresh               python
102              suresh                 java
103             kishore                    c
104              ramesh                  cpp
101*******       naresh         *********python
102*******       suresh         **********java
103*******      kishore         **************c
104*******       ramesh          ************cpp
>>>
```

**String examine methods**

**str.isspace()**

Return True if there are only whitespace characters in the string and there is at least one character, False otherwise.

```
>>> str1=''
>>> strr2=' '
>>> strr2.isspace()
True
>>>
>>> str4='python language'
>>> str4.isspace()
False
>>>
```

**str.istitle()**
Return True if the string is a titlecased string and there is at least one character, for example uppercase characters may only follow uncased characters and lowercase characters only cased ones. Return False otherwise.

```
>>> str3="Python Language"
>>> str3.istitle()
True
>>> str4="Python language"
>>> str4.istitle()
False
>>>
```

**str.isupper()**
Return True if all cased characters  in the string are uppercase and there is at least one cased character, False otherwise.

```
>>> str2="NARESH"
>>> str2.isupper()
True
>>>
```

**str.islower()**
Return True if all cased characters in the string are lowercase and there is at least one cased character, False otherwise.

```
>>> str1="naresh"
```

```
>>> str1.islower()
True
```

**str.isalnum()**
Return True if all characters in the string are alphanumeric and there is at least one character, False otherwise.

```
>>> str1="abc"
>>> str2="123"
>>> str3="abc123"
>>> str1.isalnum()
True
>>> str2.isalnum()
True
>>> str3.isalnum()
True
>>> str4="abc123$"
>>> str4.isalnum()
False
>>>
```

Example:

```
uname=input("UserName")
if uname.isalnum():
        print(f'{uname} is valid')

else:
    print(f'{uname} not allowed special character')
```

**Output:**

```
 UserNamenaresh123
naresh123 is valid
>>>
== RESTART: C:/Users/user/Desktop/python6pm/py109.py ==
UserNamenaresh$
naresh$ not allowed special character
>>>
== RESTART: C:/Users/user/Desktop/python6pm/py109.py ==
UserNamerama rao
rama rao not allowed special character
>>>
```

**str.isalpha()**
Return True if all characters in the string are alphabetic and there is at least one character, False otherwise.

```
>>> str1="naresh"
>>> str2="python3.10"
>>> str1.isalpha()
True
>>> str2.isalpha()
False
>>>
```

**String split methods**

**str.split(sep=None, maxsplit=-1)**
Return a list of the words in the string, using sep as the delimiter string. If maxsplit is given, at most maxsplit splits are done (thus, the list will have at most maxsplit+1 elements). If maxsplit is not specified or -1, then there is no limit on the number of splits (all possible splits are made).

```
str1="1 2 3 4 5"
l=str1.split()
print(l)
l=str1.split(" ")
print(l)
str2="1-2-3-4-5"
l=str2.split("-")
print(l)
l=str2.split("-",2)
print(l)
```

**str.rsplit(sep=None, maxsplit=-1)**
Return a list of the words in the string, using sep as the delimiter string. If maxsplit is given, at most maxsplit splits are done, the rightmost ones. If sep is not specified or None, any whitespace string is a separator

```
>>> str1="1 2 3 4 5"
>>> l1=str1.split()
>>> l2=str1.rsplit()
```

```
>>> print(l1)
['1', '2', '3', '4', '5']
>>> print(l2)
['1', '2', '3', '4', '5']
>>> l3=str1.split(" ",2)
>>> l4=str1.rsplit(" ",2)
>>> print(l3)
['1', '2', '3 4 5']
>>> print(l4)
['1 2 3', '4', '5']
>>>
```

**Joining methods**

**str.join(*iterable*)**
Return a string which is the concatenation of the strings in *iterable*.

```
>>> str_list=['a','b','c','d','e']
>>> str1=" ".join(str_list)
>>> print(str_list)
['a', 'b', 'c', 'd', 'e']
>>> print(str1)
a b c d e
>>> str2=",".join(str_list)
>>> print(str2)
a,b,c,d,e
>>> str1="1 2 3 4 5"
>>> l=str1.split()
>>> print(l)
['1', '2', '3', '4', '5']
>>> str2=" ".join(l)
>>> print(str2)
1 2 3 4 5
>>>
```

**Strip methods**