Joining of Strings:

join() Method

It is a string method and returns a string in which the elements of sequence have been joined by str separator.

Syntax

```
Str.join(iterable)
```

Example:

```
PyList=['1','2','3','4']
Sep="-"
print(Sep.join(PyList))
```

Example:

```
PyList=["H","E","L","L","O"]
Sep="__"
print(Sep.join(PyList))
```

Python String max() Function

It returns the max alphabetical character from the string str.

Syntax

```
max(str)
```

Example:

```
PyStr="abcd"
print(max(PyStr))
PyStr="abcddcba"
print(max(PyStr))
PyStr="Maximum"
print(max(PyStr))
```

Python String replace() Method

It returns a copy of the string in which the occurrences of old have been replaced with new, optionally restricting the number of replacements to max.

Syntax:

```
str.replace(old, new[,count])
```

Example:

```
str = "Hello How are You"
print(str.replace("You", "U"))
print(str.replace("are", "R"))
str = "Hello How are are You"
print(str.replace("are", "R"))
print(str.replace("are", "R",1))
```

title() Method

It returns a copy of the string in which first characters of all the words are capitalized.

Syntax:

```
str.title();
```

Example:

```
PyStr = "this is powerful python";
print(PyStr.title())
```

zfill() Method

It pads string on the left with zeros to fill width.

Syntax:

```
str.zfill(width)
```

Example:

```
PyStr="PYTHON"
print(PyStr.zfill(10))#0000PYTHON
print(PyStr.zfill(15))#000000000PYTHON
PyStr="MLDL"
print(PyStr.zfill(10))#000000MLDL
```

Python String isalnum() Method

It checks whether the string consists of alphanumeric characters.

Syntax

```
str.isalnum()
```

Example:

```
PyStr="2009";
print(PyStr.isalnum())
PyStr="this";
print(PyStr.isalnum())
PyStr="this2009";
print(PyStr.isalnum())
PyStr="this string example....wow"
print(PyStr.isalnum())
```

String rjust()

It returns a new string of given length after substituting a given character in left side of original string.

Syntax:

```
string.rjust(length, fillchar)
```

String ljust()

It returns a new string of given length after substituting a given character in right side of original string.

Syntax:

```
string.ljust(length, fillchar)
```

center() method

It will center align the string, using a specified character as the fill character.

Syntax:

```
center( length, fillchar )
```

Example:

```
PyStr="PYTHON"
length=10;fillchar="*"
print(PyStr.ljust(length,fillchar))
print(PyStr.rjust(length,fillchar))
print(PyStr.center(length,fillchar))
```

Removing spaces from a string:

rstrip()

It removes characters from the right based on the argument

Syntax:

```
string.rstrip([chars])
```

lstrip()

It removes characters from the left based on the argument

Syntax:

```
string.lstrip([chars])
```

strip()

It is used to remove all the leading and trailing spaces from a string.

Syntax :

```
string.strip([chars])
```

NOTE:

chars (optional) - a string specifying the set of characters to be removed.

Example:

```
PyStr='   Python For DataScience   '
print(PyStr.lstrip())
print(PyStr.rstrip())
print(PyStr.strip())
```

isspace()

It returns True if there are only whitespace characters in the string. If not, it return False.

Syntax:

```
string.isspace()
```

It returns:

True if all characters in the string are whitespace characters

False if the string is empty or contains at least one non-printable() character

Example:

```
PyStr="Python"
print(PyStr.isspace())
PyStr="Python is"
print(PyStr.isspace())
PyStr=""
print(PyStr.isspace())
```

Example:

```
PyStr="\t\t"
print(PyStr.isspace())
PyStr="\n"
```

```
print(PyStr.isspace())
PyStr=" "
print(PyStr.isspace())
```

isprintable()
It returns True if all characters in the string are printable or the string is empty. If not, it returns False.

Syntax:
```
string.isprintable()
```

Example:
```
PyStr='Space is a printable'
print(PyStr.isprintable())
PyStr='\nNew Line is printable'
print(PyStr.isprintable())
PyStr=''
print(PyStr.isprintable())
```

casefold()
It convert strings to casefolded strings for caseless matching.

Syntax:
```
string.casefold()
```

Example:
```
PyStr = "PYTHON IS AWESOME"
print(PyStr.casefold())
```

Python String swapcase()
It converts all uppercase characters to lowercase and all lowercase characters to uppercase characters of the given string, and returns it.

Syntax:
```
string.swapcase()
```

Example:
```
PyStr = "THIS SHOULD ALL BE LOWERCASE."
print(PyStr.swapcase())
PyStr="this should all be uppercase."
print(PyStr.swapcase())
PyStr= "ThIs ShOuLd Be MiXeD cAsEd."
print(PyStr.swapcase())
PyStr="UPPER lower"
print(PyStr.swapcase())
```

Python String expandtabs()
It returns a copy of string with all tab characters '\t' replaced with whitespace characters until the next multiple of tabsize parameter.

Syntax:
```
string.expandtabs(tabsize) #default tab is 4-8 spaces
```

Example:
```
PyStr='xyz\t123\tabc'
print(PyStr.expandtabs())
print(PyStr.expandtabs(10))
print(PyStr.expandtabs(15))
```

Finding Substrings:
In PYTHON programming to find sub strings we can use the following 4 methods:

Forward direction:
1 find()   2 index()

Backward direction:
1 rfind()   2 rindex()

find():
Returns index of first occurrence of the given substring. If it is not available then we will get -1

Syntax:
```
String.find(substring,begin,end)
```

Example:
```
PyStr="Learning Python is Simpler"
print(PyStr.find("Python"))
print(PyStr.find("Data"))
print(PyStr.find("e"))
```

Example:
```
PyStr="Python is"
#String.find(substring,begin,end)
print(PyStr.find('i'))
print(PyStr.find('s'))
PyStr="Python is good one"
print(PyStr.find('o'))
print(PyStr.find('o',4))
print(PyStr.find('o',5))
print(PyStr.find('o',12,16))
print(PyStr.find('o',13,16))
print(PyStr.find('O',13,16))
```

Example:
```
PyStr="hellopythonisgreat"
print(PyStr.find('a'))
print(PyStr.find('b',7,15))
print(PyStr.find('t',7,15))
print(PyStr.find('t',8,15))
```

index() method:
It returns the index of a substring inside the string (if found). If the substring is not found, it raises an exception.

Syntax:
```
str.index(sub[, start[, end]] )
```

Example:
```
PyStr='Python programming is fun'
print(PyStr.index('is fun'))
print(PyStr.index('ing', 10))
print(PyStr.index('g is', 10, -4))
```