Python Strings with Examples

Strings in Python are sequences of characters that can be easily created. Python Strings are objects of **str** class. Let's see what a string is,

```
"My name is Kevin" – This is a string
```

Always remember that in Python, single quotes are similar to double quotes. However, triple quotes are also used, but for multi-line string.

Create a String in Python

This is how you can create a string in Python:

```
str = 'My name is Kevin!'
```

String Literals

In Python Strings, literals are enclosed in single, double quotes, and even triple quotes, for example:

```
#!/usr/bin/python

str1 = 'My name is Kevin!'
print (str1)

str2 = "My name is Kevin!"
print (str2)

#multi-line string
str3 = """My name is Kevin and I
          live in New York"""
print (str3)
```

The output is as follows:

```
My name is Kevin!
```

```
My name is Kevin!
My name is Kevin and I
          live in New York
```

Above, under variable **str3**, you can see the string surrounded by three quotes. This forms the string, multi-line.

Slicing to access Substrings

Access characters in a range with Slicing in Python Strings. Let's say you only want to access substring "**Kevin**", for that, we use square brackets. Mention the start index and the end index, separate by colon, to get substring in a range.

For accessing a substring in Python Strings,

```
myVar = 'My name is Kevin!'
print "Here's my name: ", myVar[11:16]
```

The output is,

```
Here's my name:  Kevin
```

Negative Indexing to access substrings

If you want to begin slicing from the end of the string, use Negative Indexing. Let us see an example:

```
str = "This is Demo Text"
print("String = ",str)

# Negative Indexing
print("Accessing substring with negative indexing...")
print(str[-4:-2])
print(str[-6:-2])
print(str[-7:-5])
```

The output is as follows:

```
String = This is Demo Text
Accessing substring with negative indexing...
Te
o Te
mo
```

Accessing a Character in Python Strings

Access a Character from Python Strings is quite easy. Let's say you only want to access substring "s", for that, we use square brackets. In some programming languages like C, C++, etc, we have a character type. Under Python Strings, we do not have character type, so strings of length one is considered a character.

For accessing a string character,

```
myVar = 'My name is Kevin!'
print "Character is: ", myVar[9]
```

The output is,

```
Character is: s
```

Escape Characters in Python Strings

The escape characters are used to insert characters, which are not allowed. Use backslash \, since it is used as an escape character. Follow it by the character to be inserted.

Following are the escape characters used in Python:

Notation	Description
\a	Bell or alert
\b	Backspace
\e	Escape
\f	Formfeed
\n	Newline
\nnn	Octal notation, where n is in the range 0-7
\r	Carriage Return
\s	Space
\t	Tab
\v	Vertical Tab
\x	Character x
\xnn	Hexadecimal notation, where n is in the range 0-9, a-f, or A-F

Let us now see some examples of the Escape Characters in

Python:

Examples of Escape Characters in Python

Following is an example displaying the usage of Escape Characters in Python:

```
## Escape Characters

# Form Feed
str = "This is \fDemo Text"
print("String = ",str)

# Octal Notation
str = '\110'
print("Octal Notation = ",str)

# Hexadecimal Notation
str = '\x57'
print("Hexadecimal Notation = ",str)

# Tab Notation
str = 'Demo\tText'
print("Hexadecimal Notation = ",str)
```

The output is as follows:

```
String = This is Demo Text
Octal Notation = H
Hexadecimal Notation = W
Hexadecimal Notation = Demo      Text
```

Concatenate String in Python

Use the + operator to concatenate String in Python. Let us see an example:

```
## Concatenate Strings

# String1
str1 = "Money"
print("String 1 = ",str1)

# String2
str2 = "Heist"
print("String 2 = ",str2)

# concat
res = str1 + str2

print("Complete String = ",res)
```

The output is as follows:

```
String 1 = Money
String 2 = Heist
Complete String = MoneyHeist
```

String Operators

Python Strings comes with special Operators. Below are these operators, let's say, we have two variables, with values:

```
X = Study
Y = Now
```

Following are the String operators:

Operator	Description	Sample
+	Concatenates two strings	X+ Y 'StudyNow'
*	Concatenates copies of the same string	X*2 'StudyStudy'
[]	Returns the character from the given index	X[2] u
[:]	Gets the characters from a given range	X[0:3] 'Stud'
in	It returns true if a character exists in the given string	Finding 'w' in variable Y True
not in	It returns true if a character isn't in the given string	Finding 'w' in variable Y False Finding 'K' in variable X True

Python Built-in String Methods

Let us work on Python Strings Methods:

Python capitalize() method

The capitalize() method in Python, as the name suggests, is used to capitalize (uppercase) only the first letter of sentence. Let us see an example:

```
# capitalize method

str = "captain philips"
print("Initial string = ",str);

res = str.capitalize()

print("Updated string = ",res);
```

The output is as follows:

```
Initial string =  captain philips
Updated string =  Captain philips
```

Let us see another example to implement the capitalize() method:

```
# capitalize method

str = "rON wEASLEY"
print("Initial string = ",str);

res = str.capitalize()

print("Updated string = ",res);
```

The output is as follows:

```
Initial string =  rON wEASLEY
Updated string =  Ron Weasley
```

Python casefold() method

The casefold() method in Python is used to convert a string into lowercase. Let us see an example:

```
# casefold method

str = "CAPTAIN PHILIPS"
print("Initial string = ",str);

res = str.casefold()

print("Updated string = ",res);
```

The output is as follows:

```
Initial string =  CAPTAIN PHILIPS
Updated string =  captain philips
```

Let us see another example to implement the casefold() method:

```
# casefold method

str = "Gilderoy Lockhart"
print("Initial string = ",str);

res = str.casefold()

print("Updated string = ",res);
```

The output is as follows:

```
Initial string =  Gilderoy Lockhart
Updated string =  gilderoy lockhart
```

Python center() method

The center() method in Python is used to align a string centrally. You can also use any character as a fill character each size, but this is optional. The syntax is as follows:

```
string.center(len, char)
```

Above, len is the length of the string, whereas, char is the character to filled on left and right side.

Let us see an example to implement the center() method:

```
# center method

str = "DEMO"
print("Initial string = ",str);

res = str.center(10,"#")

print("Updated string = ",res);
```

The output is as follows:

```
Initial string =  DEMO
Updated string =  ###DEMO###
```

```
Updated string =  ###DEMO###
```

Let us see another example to implement the center() method:

```
# center method

str = "TEST0.1"
print("Initial string = ",str);

res = str.center(10,"$")

print("Updated string = ",res);
```

The output is as follows:

```
Initial string =  TEST0.1
Updated string =  $TEST0.1$$
```

Python count() method

The count() method in Python is used to search for a specific value and its count of appearance in the string. The syntax is as follows:

```
string.count(str, begn, end)
```

Above, **str** is the string, **begn** is the position to start the search and end is the position to end.

Let us see an example to implement the count() method on Python:

```
# count method

str = "This is one."
print("String = ",str);

res = str.count("This",0, 15)
print("Count of specific value = ",res);
```

The output is as follows:

```
String =  This is one.
Count of specific value =  1
```

Let us now see another example to implement the count() method:

```
# count method

str = "This is demo. This is another demo."
print("String = ",str);

res = str.count("This")
print("Count of specific value = ",res);
```

The output is as follows:

```
String =  This is demo. This is another demo.
Count of specific value =  2
```

Python encode() method

The encode() method in Python encodes the string. Default is UTF-8. Let us see an example:

```
# encode method

str = "This is årea."
print("String = ",str);

res = str.encode()
print("Encoded String = ",res);
```

The output is as follows:

```
String =  This is årea.
Encoded String =  b'This is \xc3\xa5rea.'
```

Python find() method

The find() method in Python is used to search for the occurrence of a specific letter in a string. The syntax is as follows:

```
string.find(str, begin, end)
```

Above, **str** is the string to be searched for, **begin** is where to

start the search and **end** is where the search ends. Let us see an example:

```
# find method

str = "This is first text."
print("String = ",str);

res = str.find("text")

print("The string has a specific value at position = ",res);
```

The output is as follows:

```
String = This is first text.
The string has a specific value at position = 14
```

Let us see another example to implement the find() method:

```
# find method

str = "This is first text. This is second text"
print("String = ",str);

res = str.find("text", 20, 40)
print("The string has a specific value at position = ",res);
```

The output is as follows:

```
String = This is first text. This is second text
The string has a specific value at position = 35
```

Python endswith() method

The endswith() method in Python is used to check for a string. If this string ends with the specified value, TRUE is returned, else False. The syntax is as follows:

```
string.endswith(str, begin, end)
```

Above, **str** is the value to be checked if the string ends with, **begin** is the position where the search start, whereas **end** is where the search ends.

Let us see an example to implement the endswith() method:

```
# endswith method

str = "This is demo text. This is demo text2"
print("String = ",str);

res = str.endswith("text2")
print("The string ends with a specific value = ",res);
```

The output is as follows:

```
String = This is demo text. This is demo text2
The string ends with a specific value = True
```

Let us see another example of the endswith() method:

```
# endswith method

str = "This is demo text. This is demo text2"
print("String = ",str);

res = str.endswith("text2", 7, 15)
print("The string ends with a specific value from position 7 to
```

The output is as follows:

```
String = This is demo text. This is demo text2
The string ends with a specific value from position 7 to 15 =
```

Python expandtabs() method

The expandtabs() method in Python is used to set the tab size. The default tab size, which we all must have used is 8. To set a different tab size i.e. number of whitespaces, use the expandtabs().

Let us see an example, wherein we will set the tab size as the parameter if the expandtabs() method:

```
# expandtabs method

str = "D\te\tm\to\t"
print("Initial string (with default tab size 8) = ",str.expandt

print("String (with tab size 2) = ",str.expandtabs(2));
print("String (with tab size 4) = ",str.expandtabs(4));
```

The output is as follows:

```
Initial string (with default tab size (8) = D       e       m
String (with tab size 2) = D e m o
String (with tab size 4) = D   e   m   o
```

Let us see another example to implement the expandtabs() method:

```
# expandtabs method

str = "Test\tTest2"

print("String (with tab size 10) = ",str.expandtabs(10));
print("String (with tab size 0) = ",str.expandtabs(0));
```

The output is as follows:

```
String (with tab size 10) = Test      Test2
String (with tab size 0) =  TestTest2
```

Python index() method

Use the index() method in Python to find the index of the first occurrence of a specific value. Let us see the syntax:

```
string.index(str, begin, end)
```

Above, **str** is the value to be searched, begin is where the search begins and end is where it ends.

Let us now see an example of index() method in Python:

```
# index method

str = "This is first text."
print("String = ",str);

res = str.index("i")
print("The first occurrence of the specific value found at inde
```

The output is as follows:

```
String = This is first text.
The first occurrence of the specific value found at index = 2
```

Let us see another example of index() method with search beginning from a specific index:

```
# index method

str = "This is first text."
print("String = ",str);

res = str.index("i", 3, 10)
print("The first occurrence of the specific value found at inde
```

The output is as follows:

```
String = This is first text.
The first occurrence of the specific value found at index begin
```

Python isalnum() method

The isalnum() method in Python returns TRUE, if all the characters in the string are alphanumeric, else FALSE is returned.

Let us see an example of the isalnum() method:

```
# isalnum method

str = "JamesBond007"
print("String = ",str);

res = str.isalnum()
print("Are all the characters in the string alphanumeric = ",re
```

The output is as follows:

```
String = JamesBond007
Are all the characters in the string alphanumeric = True
```

Let us see another example to implement the isalnum() method in Python:

```
# isalnum method
```

```
str = "$$$$###55KP"
print("String = ",str);

res = str.isalnum()

print("Are all the characters in the string alphanumeric = ",re
```

The output is as follows:

```
String =  $$$$###55KP
Are all the characters in the string alphanumeric =  False
```

Python isalpha() method

The isalpha() method in Python returns TRUE, if all the characters in the string are alphabet, else FALSE is returned.

Let us see an example of the isalpha() method:

```
# isalpha method

str1 = "JamesBond007"
print("String1 = ",str1);

res = str1.isalnum()
print("Are all the characters in String1 alphanumeric = ",res);

str2 = "TomCruise"
print("String2 = ",str2);

res = str2.isalpha()
print("Are all the characters in String2 alphabets = ",res);
```

The output is as follows:

```
String1 = JamesBond007
Are all the characters in String1 alphanumeric = True
String2 = TomCruise
Are all the characters in String2 alphabets = True
```

Let us see another example to implement the isalpha() method:

```
# isalpha method

str1 = "@@Jack"
print("String1 = ",str1);

res = str1.isalnum()
print("Are all the characters in String1 alphanumeric = ",res);

str2 = "$$$$$K"
```

```
print("String2 = ",str2);

res = str2.isalpha()
print("Are all the characters in String2 alphabets = ",res);
```

The output is as follows:

```
String1 =  @@Jack
Are all the characters in String1 alphanumeric =  False
String2 =  $$$$$K
Are all the characters in String2 alphabets =  False
```

Python isdecimal() method

The isdecimal() method in Python returns TRUE, if all the characters in the string are decimals, else FALSE is returned.

Let us see an example of the isdecimal() method:

```
# isdecimal method

str = "37"
print("String = ",str);

res = str.isdecimal()
print("Are all the characters in String (Unicode) are decimal =
```

The output is as follows:

```
String =  37
Are all the characters in String (Unicode) are decimal =  True
```

Python isdigit() method

The isdigit() method in Python returns TRUE, if all characters in the string are digits, else FALSE is returned.

Let us see an example of the isdigit() method:

```
# isdigit method

str = "47890"
print("String = ",str);

res = str.isdigit()
print("Are all the characters in String are digits = ",res);
```

The output is as follows:

```
String =  47890
Are all the characters in String are digits = True
```

Let us see another example of the isdigit() method:

```
# isdigit method

str = "demo666"
print("String = ",str);

res = str.isdigit()
print("Are all the characters in String are digits = ",res);
```

The output is as follows:

```
String =  demo666
Are all the characters in String are digits = False
```

Python isidentifier() method

The isidentifier() method in Python returns TRUE, if the string is an identifier, else FALSE is returned.

Note: Valid identifier doesn't begin with a number. It doesn't even include spaces. However, a string is a valid identifier, if it includes (a-z), (0-9), or (_).

Let us see an example of the isidentifier() method:

```
# isidentifier method

str = "demo666"
print("String = ",str);

res = str.isidentifier()
print("Is the string, a valid identifier = ",res);
```

The output is as follows:

```
String =  demo666
Is the string, a valid identifier =  True
```

Let us see another example to implement isidentifier() method:

```
# isidentifier method

str = "38768"
print("String = ",str);

res = str.isidentifier()
print("Is the string, a valid identifier = ",res);
```

The output is as follows:

```
String =  38768
Is the string, a valid identifier =  False
```

Python islower() method

The islower() method in Python returns TRUE, if all the characters in the string are lowercase, else FALSE is returned.

Let us see an example of the islower() method:

```
# islower method

str = "jackryan"
print("String = ",str);

res = str.islower()
print("Are all the characters in the string lowercase = ",res);
```

The output is as follows:

```
String =  jackryan
Are all the characters in the string lowercase =  True
```

Let us see another example to implement the islower() method:

```
# islower method

str = "JackRyan"
print("String = ",str);

res = str.islower()
print("Are all the characters in the string lowercase = ",res);
```

The output is as follows:

```
String =  JackRyan
Are all the characters in the string lowercase =  False
```

Python isnumeric() method

The isnumeric() method in Python returns TRUE, if all the characters in the string are numeric, else FALSE is returned.

Let us see an example of the isnumeric() method:

```
# isnumeric method

str = "987987"
print("String = ",str);

res = str.isnumeric()
print("Are all the characters in the string numeric = ",res);
```

The output is as follows:

```
String =  987987
Are all the characters in the string numeric =  True
```

Let us now see another example to implement the isnumeric() method:

```
# isnumeric method

str = "Sylvester"
print("String = ",str);

res = str.isnumeric()
print("Are all the characters in the string numeric = ",res);
```

The output is as follows:

```
String =  Sylvester
Are all the characters in the string numeric =  False
```

Python isprintable() method

The isprintable() method in Python returns TRUE, if all the characters in the string are alphabet, else FALSE is returned.

Let us see an example of the isprintable() method:

Let us see an example of the isprintable() method:

```
# isprintable method

str = "Sylvester"
print("String = ",str);

res = str.isprintable()
print("Are all the characters in the string printable = ",res);
```

The output is as follows:

```
String =  Sylvester
Are all the characters in the string printable =  True
```

Let us see another example to implement the isprintable() method:

```
# isprintable method

str = "Sylvester\tStallone"
print("String = ",str);

res = str.isprintable()
print("Are all the characters in the string printable = ",res);
```

The output is as follows:

```
String =  Sylvester Stallone
Are all the characters in the string printable =  False
```

Python isspace() method

The isspace() method in Python returns TRUE, if all the characters in the string are whitespaces, else FALSE is returned.

Let us see an example of the isspace() method:

```
# isspace method

str = "   "
print("String = ",str);

res = str.isspace()
print("Are all the characters in the string whitespace = ",res)
```

The output is as follows:

```
String =
Are all the characters in the string whitespace =  True
```

Let us see another example to implement the isspace() method:

```
# isspace method

str = "One  Two     Three"
print("String = ",str);

res = str.isspace()
print("Are all the characters in the string whitespace = ",res)
```

The output is as follows:

```
String =  One  Two     Three
Are all the characters in the string whitespace =  False
```

Python istitle() method

The istitle() method in Python eturns TRUE, if all the string considers the rules of a title (i.e. uppercase title), else FALSE is returned. Uppercase titles, example, “Demo Text”, “Mark”, etc.

Let us see an example of the istitle() method:

```
# istitle method

str = "Mark"
print("String = ",str);

res = str.istitle()
print("Is the string considers the Uppercase title rule = ",res
```

The output is as follows:

```
String =  Mark
Is the string considers the Uppercase title rule =  True
```

Let us see another example to implement the istitle() method:

```
# istitle method

str = "demo text"
print("String = ",str);
```

```
res = str.istitle()
print("Is the string considers the Uppercase title rule = ",res
```

The output is as follows:

```
String =  demo text
Is the string considers the Uppercase title rule =  False
```

Python isupper()method

The isupper() method in Python returns TRUE, if all the characters in the string are uppercase, else FALSE is returned.

Let us see an example of the isupper() method:

```
# isupper

str = "MARK RUFFALO"
print("String = ",str);

res = str.isupper()
print("Are all the characters in the string uppercase =",res);
```

The output is as follows:

```
String = MARK RUFFALO
Are all the characters in the string uppercase = True
```

Let us see another example to implement the isupper() method:

```
# isupper

str = "demo TEXT"
print("String = ",str);

res = str.isupper()
print("Are all the characters in the string uppercase =",res);
```

The output is as follows:

```
String =  demo TEXT
Are all the characters in the string uppercase = False
```

Python join() method

The join() method in Python join is used to join the iterable elements to the end of the string. Containers, such as Tuple, String, Dictionary, etc. are iterables in Python.

Let us see an example of the join() method, wherein we have Dictionary iterable, returning keys:

```
# join method

dict = {"id": "S01", "name": "Frank"}
print("Dictionary = ",dict)

# set separator
sep = "AAA"

res = sep.join(dict)
print("Dictionary iterable =", res)
```

The output is as follows:

```
Dictionary =  {'name': 'Frank', 'id': 'S01'}
Dictionary iterable = nameAAAid
```

Let us see an example of the Strings iterable with Python join():

```
# join method

str = {"Frank", "Shaun", "Scarlett", "Morgan"}
print("String = ",str)

# set separator
sep = "$$"

res = sep.join(str)
print("String iterable =", res)
```

The output is as follows:

```
String = {'Scarlett', 'Shaun', 'Frank', 'Morgan'}
String iterable = Scarlett$$Shaun$$Frank$$Morgan
```

Python lower() method

The lower() method in Python is used to convert the uppercase letters in the string to lowercase.

Let us see an example of the lower() method:

```
# lower method

str = "The Walking Dead"
print("String = ",str)

res = str.lower()

print("Updated String with all letters in lowercase =", res)
```

The output is as follows:

```
String =  The Walking Dead
Updated String with all letters in lowercase = the walking dead
```

Let us see another example to implement the lower() method in Python:

```
# lower method

str = "STRANGER THINGS"
print("String = ",str)

res = str.lower()

print("Updated String =", res)
```

The output is as follows:

```
String = STRANGER THINGS
Updated String = stranger things
```

Python lstrip() method

The lstrip() method in Python is used to remove characters from the left of the string. Default is whitespace. The syntax is as follows:

```
lstrip(ch)
```

Above, the parameter **ch** are the characters to be trimmed.

Let us see an example of the lstrip() method:

```
# lstrip method

str = "######STRANGER THINGS"
print("String = ",str)

res = str.lstrip("#")

print("Updated String =", res)
```

The output is as follows:

```
String =       TRUE DETECTIVE
Updated String = TRUE DETECTIVE
```

Python replace() method

The replace() method in Python is used to replace a value with another. The syntax is as follows:

```
replace(old, new, count)
```

Above, old is the value to be searched, new is the value to be replaced with old, and count is how many occurrences of the old value to be replaced.

Let us see another example:

```
# replace method

str = "This is demo. This is another demo"
print("String = ",str)

res = str.replace("demo", "test", 2)

print("Updated String =", res)
```

The output is as follows:

```
String = This is demo. This is another demo
Updated String = This is test. This is another test
```

Let us see another example to implement the replace() method:

```
# replace method

str = "This is demo."
```

```
print( String =  ,str)

res = str.replace("demo", "example")

print("Updated String =", res)
```

The output is as follows:

```
String =  This is demo.
Updated String = This is example.
```

Python rfind() method

The rfind() method in Python is used to find the last occurrence of a value from a string. The syntax is as follows:

```
rfind(val, begin, end)
```

Above, val is the value to be searched, begin is from where the search begins, and end is where it ends.

Let us see an example to implement the rfind() method:

```
# rfind method

str = "This is demo. This is another demo"
print("String = ",str)

res = str.rfind("demo")

print("Position of Last occurrence of a specific value =", res)
```

The output is as follows:

```
String = This is demo. This is another demo
Position of Last occurrence of a specific value = 30
```

Let us see another example to implement the rfind() method:

```
# rfind method

str = "This is demo. This is another demo. Yet another demo."
print("String = ",str)

res = str.rfind("demo")

print("Position of Last occurrence a specific value =", res)
```

The output is as follows:

```
String = This is demo. This is another demo. Yet another demo.
Position of Last occurrence a specific value = 48
```

Python rindex()method

The rindex() method in Python is used to get the index of the last occurrence of a specific value. The syntax is as follows:

```
rindex(val, begin, end)
```

Above, val is the value to be searched, begin is from where the search begins, and end is where it ends.

Note: rindex() is the same as rfind()

Let us see an example to implement the rindex() method in Python:

```
# rindex method

str = "This is test. This is another test"
print("String = ",str)

res = str.rindex("test")

print("Position of Last occurrence of a specific value =", res)
```

The output is as follows:

```
String = This is test. This is another test
Position of Last occurrence of a specific value = 30
```

Let us see another example to implement the rindex() method:

```
# rindex method

str = "Test test test test test test test test"
print("String = ",str)

res = str.rindex("test")

print("Position of Last occurrence of a specific value =", res)
```

The output is as follows:

```
String =  Test test test test test test test test
Position of Last occurrence of a specific value = 35
```

Python rjust() method

The rjust() method in Python is used to right align the string and fill characters to the left. The syntax is as follows:

```
rjust(len, chars)
```

Above, **len** is the total length of the string to be returned, whereas chars are the characters to be files on the left.

Let us see an example to implement the rjust() method in Python:

```
# rjust method

str = "American Vandal"
print("String = ",str)

res = str.rjust(25, "A")

print(res)
```

The output is as follows:

```
String =  American Vandal
AAAAAAAAAAAmerican Vandal
```

Let us see another example to implement the rjust() method:

```
# rjust method

str = "American"
print("String = ",str)

res = str.rjust(15, "A")

print(res, "Vandal")
```

The output is as follows:

```
String =  American
AAAAAAAAAAAAAAAAAAmerican Vandal
```

Python rsplit() method

The rsplit() method in Python is used to split a string, beginning from the right. A list is returned. The syntax is as follows:

```
rsplit(sep, split)
```

Above, sep is the separator used while splitting, and split is the number of splits to be performed. If you won't mention anything, that would mean "all occurrences".

Let us see an example to implement the rsplit() method in Python:

```
# rsplit method

str = "Football, Archery, Cricket, Squash, Hockey, Volleyball"
print("String = ",str)

# split returns a list with 2 elements
res = str.rsplit(", ", 2)

print(res)
```

The output is as follows:

```
String = Football, Archery, Cricket, Squash, Hockey, Volleyball
['Football, Archery, Cricket, Squash', 'Hockey', 'Volleyball']
```

Note: Above, since we mentioned 2 as a parameter for split, the output is two elements i.e.:

```
Element 1: 'Football, Archery, Cricket, Squash'
Element 2: 'Hockey', 'Volleyball'
```

Let us see another example to implement the rsplit() method in Python:

```
# rsplit method

str = "Football, Archery, Cricket, Squash, Hockey, Volleyball"
print("String = ",str)

# return a comma separated list
res = str.rsplit(", ")

print(res)
```

The output is as follows:

```
String = Football, Archery, Cricket, Squash, Hockey, Volleyball
['Football', 'Archery', 'Cricket', 'Squash', 'Hockey', 'Volleyb
```

Python rstrip() method

The rstrip() method in Python is used to remove characters from the right of the string. Default is whitespace. The syntax is as follows:

```
rstrip(ch)
```

Above, the parameter ch are the characters to be trimmed.

Let us see an example of the rstrip() method:

```
# rstrip method

str = "STRANGER THINGS#######"
print("String = ",str)

res = str.rstrip("#")

print("Updated String =", res)
```

The output is as follows:

```
String = STRANGER THINGS#######
Updated String = STRANGER THINGS
```

Let us see another example to implement the rstrip() method and remove whitespace from the right:

```
# rstrip method

str = "TRUE DETECTIVE                    "
print("String = ",str)

res = str.rstrip()

print("Updated String =", res)
```

The output is as follows:

```
String = TRUE DETECTIVE
Updated String = TRUE DETECTIVE
```

Python split() method

The split() method in Python is used to split a string into a list. You can also set the separator. Whitespace is the default separator. The syntax is as follows:

```
split(sep, split)
```

Above, **sep** is the separator used while splitting, and split is the number of splits to be performed. If you won't mention anything, that would mean "**all occurrences**".

Let us see an example of the split() method, wherein 2 elements are returned after split:

```
# split method

str = "One$$Two$$Three$$Four"
print("String = ",str)

res = str.split("$$",1)

print("Result =", res)
```

The output is as follows:

```
String =  One$$Two$$Three$$Four
Result = ['One', 'Two$$Three$$Four']
```

Note: Above, since we mentioned 2 as a parameter for split, the

output is two elements i.e.:

```
Element 1: 'One'
Element 2: 'Two$$Three$$Four'
```

Python splitlines() method

The splitlines() method in Python is used to split a string into list. You can also set option to allow line breaks or not. The syntax is as follows:

```
string.plitlines(linebrk)
```

Above, **linebrk** is a boolean value to set whether to allow line breaks or not.

Let us see an example to implement the splitlines() method:

```
# splitlines method
str = "One\nTwo\nThree\nFour"
res = str.splitlines(True)
print("Result =", res)
```

The output is as follows:

```
Result = ['One\n', 'Two\n', 'Three\n', 'Four']
```

Let us see another example to implement the splitlines() method:

```
# splitlines method
str = "One\nTwo\nThree\nFour"
res = str.splitlines(True)
print("Result =", res)
```

The output is as follows:

```
Result = ['One', 'Two', 'Three', 'Four']
```

Let us see another example to implement the splitlines() method:

```
# splitlines method
str = "One\nTwo\nThree\nFour"
res = str.splitlines(False)
print("Result =", res)
```

The output is as follows:

```
Result = ['One', 'Two', 'Three', 'Four']
```

Python startswith() method

The startswith() method in Python returns TRUE, if string begins with a particular value., else FALSE is returned. The syntax is as follows:

```
string.startswith(val, begin, end)
```

Above, **val** is the value to be checked where the string begins with, **begin** is the position where the search begins, and **end** is the position where the search ends.

Let us see an example to implement the startswith() method in Python:

```
# startswith method
str = "This is demo. This is another demo."
print("String = ",str)
res = str.startswith("other", 15, 25)
print("Does any word begin with other =", res)
```

The output is as follows:

```
String = This is demo. This is another demo.
Does any word begin with other = False
```

Let us see another example to implement the startswith() method:

```
# startswith method

str = "This is demo. This is another demo."
print("String = ",str)

res = str.startswith("an", 22, 30)

print("Does any word begin with the an =", res)
```

The output is as follows:

```
String = This is demo. This is another demo.
Does any word begin with the an = True
```

Python swapcase() method

The swapcase() method in Python is used swap the case i.e. convert lowercase to uppercase, and vice versa.

Let us see an example to implement the swapcase() method in Python:

```
# swapcase method

str = "This is demo. This is another demo."
print("String = ",str)

res = str.swapcase()

print("Case Swapped =", res)
```

The output is as follows:

```
String = This is demo. This is another demo.
Case Swapped = tHIS IS DEMO. tHIS IS ANOTHER DEMO.
```

Let us see another example to implement the swapcase() method in Python:

```
# swapcase method

str = "STRANGER THINGS"
print("String = ",str)

res = str.swapcase()

print("Case Swapped =", res)
```

The output is as follows:

```
String = STRANGER THINGS
Case Swapped = stranger things
```

Python title() method

The title() method in Python is used to convert the first letter of every work lettercase i.e. title, heading, etc.

Let us see an example to implement the title() method in Python:

```
# title method

str = "STRANGER THINGS"
print("String = ",str)

res = str.title()

print("Updated String =", res)
```

The output is as follows:

```
String = STRANGER THINGS
Updated Strings = Stranger Things
```

Let us see another example to implement the title() method:

```
# title method

str = "demo text"
print("String = ",str)

res = str.title()

print("Updated String =", res)
```

The output is as follows:

```
String = demo text
Updated String = Demo Text
```

Python upper() method

Use the upper() method in Python to convert the lowercase letters in the string to uppercase.

Let us see an example of the upper() method:

```
# upper method

str = "The Walking Dead"
print("String = ",str)

res = str.upper()

print("Updated String with all letters in uppercase now =", res
```

The output is as follows:

```
String = The Walking Dead
Updated String with all letters in uppercase now = THE WALKING
```

Let us see another example to implement the upper() method in Python:

```
# upper method

str = "demo text"
print("String = ",str)

res = str.upper()

print("Updated String with all letters in uppercase now =", res
```

The output is as follows:

```
String = demo text
Updated String with all letters in uppercase now = DEMO TEXT
```

Python zfill() method

The zfill() method in Python is use to fill zeros in the beginning of the string. The syntax is as follows:

```
zfill(len)
```

Above, the parameter len is the total length of the string including the zero fill.

Let us see an example to implement the zfill() method in Python:

```
# zfill method

str = "Wow!"
print("String = ",str)

res = str.zfill(7)

print("Updated String with zero fill =", res)
```

The output is as follows:

```
String = Wow!
Updated String with zero fill = 000Wow!
```

Let us see another example to implement the zfill() method:

```
# zfill method

str = "155.898"
print("String = ",str)

res = str.zfill(15)

print("Updated String with zero fill =", res)
```

The output is as follows:

```
String =  155.898
Updated String with zero fill = 00000000155.898
```

Python strip() method

The strip() method in Python is used to remove spaces from the beginning and end of a string. The syntax is as follows:

```
string.strip(ch)
```

```
.strip(ch)
```

Above, the parameter ch are the set of characters to be removed from the beginning and end of the string.

Let us see an example to implement the strip() method in Python:

```
# strip method

str = "      10.50  "
print("String = ",str)

res = str.strip()

print("Updated String by removing leading and trailing whitespa
```

The output is as follows:

```
String =        10.50
Updated String by removing leading and trailing whitespaces = 1
```

Let us see another example to implement the strip() method in Python and removing leading and trailing characters:

```
# strip method

str = "$,...jack...$"
print("String = ",str)

res = str.strip("$,.")

print("Updated String by removing leading and trailing characte
```

The output is as follows:

```
String =  $,...jack...$
Updated String by removing leading and trailing characters = ja
```

In this guide, we saw how to work with Python Strings, accessing a specific character, escape characters and the built-in string methods.

Read More

- Functions in Python
- Scope of Variables
- Type Conversion
- Python Lists
- Python DateTime
- Tuples in Python
- Python Dictionary