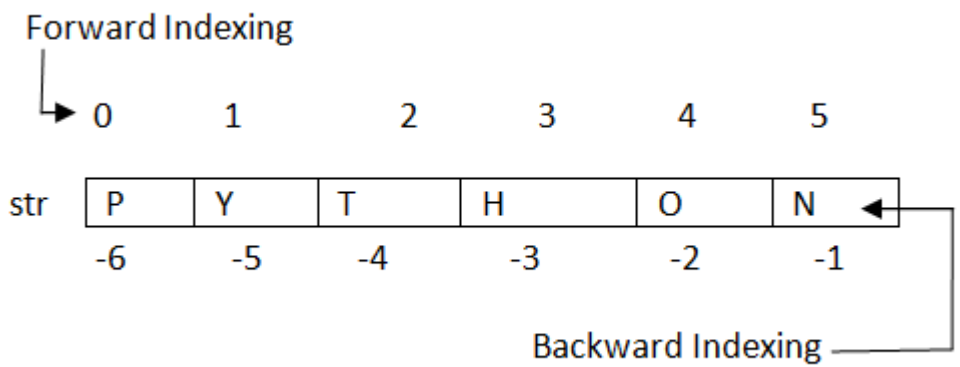
PYTHON STRINGS

- Strings are the simplest and easy to use in Python.
- String pythons are immutable.
- We can simply create Python String by enclosing a text in single as well as double quotes. Python treat both single and double quotes statements same.

Accessing Strings:

- In Python, Strings are stored as individual characters in a contiguous memory location.
- The benefit of using String is that it can be accessed from both the directions in forward and backward.
- Both forward as well as backward indexing are provided using Strings in Python.
- Forward indexing starts with 0,1,2,3,....
- Backward indexing starts with -1,-2,-3,-4,....

Eg:



Strings Operators

There are basically 3 types of Operators supported by String:

- Basic Operators.
- Membership Operators.
- Relational Operators.

Basic Operators:

There are two types of basic operators in String. They are "+" and "*".

String Concatenation Operator :(+)

The concatenation operator (+) concatenate two Strings and forms a new String.

eg:

```
>>> "ratan" + "jaiswal"
Output:
'ratanjaiswal'
```

NOTE: Both the operands passed for concatenation must be of same type, else it will show an error.

Eg:

```
'abc' + 3    TypeError: cannot concatenate 'str' and 'int' objects
```

Replication Operator: (*)

Replication operator uses two parameter for operation. One is the integer value and the other one is the String.

The Replication operator is used to repeat a string number of times. The string will be repeated the number of times which is given by the integer value.

Eg:

```
>>> 5*"Vimal"
```

Output:

```
'VimalVimalVimalVimalVimal'
```

```
NOTE: We can use Replication operator in any way i.e., int * string
or string * int. Both the parameters passed cannot be of same type.

>>>'$'*5
Output
'$$$$$'
```

Membership Operators

There are two types of Membership operators:

1) in:"in" operator return true if a character or the entire substring is present in the specified string, otherwise false.

2) not in:"not in" operator return true if a character or entire substring does not exist in the specified string, otherwise false.

Eg:

```
>>> str1="javatpoint"
>>> str2='sssit'
>>> str3="seomount"
>>> str4='java'
>>> st5="it"
>>> str6="seo"
>>> str4 in str1
True
```

```
>>> str5 in str2
>>> st5 in str2
True
>>> str6 in str3
True
>>> str4 not in str1
False
>>> str1 not in str4
True
```

Relational Operators:

All the comparison operators i.e., (<,><=,>=,==,!=,<>) are also applicable to strings.

The Strings are compared based on the ASCII value or Unicode(i.e., dictionary Order).

Eg:

```
>>> "RAJAT"=="RAJAT"
True
>>> "alisha">='Alisha'
True
>>> "Z"<>"z"
True
```

Slice Notation:

String slice can be defined as substring which is the part of string. Therefore further substring can be obtained from a string.

There can be many forms to slice a string. As string can be accessed or indexed from both the direction and hence string can also be sliced from both the direction that is left and right.

Syntax:

<string_name>[startIndex:endIndex],

<string_name>[:endIndex],

<string_name>[startIndex:]

Example:

```
>>> str="Nikhil"
>>> str[0:6]
'Nikhil'
>>> str[0:3]
'Nik'
>>> str[2:5]
'khi'
```

```
>>> str[:6]
'Nikhil'
>>> str[3:]
'hil'
```

Note: startIndex in String slice is inclusive whereas endIndex is exclusive.

String Functions and Methods:

capitalize()	It capitalizes the first character of the String.
count(string,begin,end)	Counts number of times substring occurs in a String between begin and end index.
endswith(suffix ,begin=0,end=n)	Returns a Boolean value if the string terminates with given suffix between begin and end.
find(substring ,beginIndex, endIndex)	It returns the index value of the string where substring is found between begin index and end index.
index(subsring, beginIndex, endIndex)	Same as find() except it raises an exception if string is not found.
isalnum()	It returns True if characters in the string are alphanumeric i.e., alphabets or numbers and there is at least 1 character. Otherwise it returns False.
isalpha()	It returns True when all the characters are alphabets and there is at least one character, otherwise False.
isdigit()	It returns True if all the characters are digit and there is at least one character, otherwise False.

islower()	It returns True if the characters of a string are in lower case, otherwise False.
isupper()	It returns False if characters of a string are in Upper case, otherwise False.
isspace()	It returns True if the characters of a string are whitespace, otherwise false.
len(string)	len() returns the length of a string.
lower()	Converts all the characters of a string to Lower case.
upper()	Converts all the characters of a string to Upper Case.
startswith(str ,begin=0,end=n)	Returns a Boolean value if the string starts with given str between begin and end.
swapcase()	Inverts case of all characters in a string.
lstrip()	Remove all leading whitespace of a string. It can also be used to remove particular character from leading.
rstrip()	Remove all trailing whitespace of a string. It can also be used to remove particular character from trailing.

1) capitalize()

>>> 'abc'.capitalize()

Output:

```
'Abc'
```

2) count(string)

```
msg = "welcome to sssit";
substr1 = "o";
print  msg.count(substr1, 4, 16)
substr2 = "t";
print  msg.count(substr2)
Output:
>>>
2
2
```

3) isspace()

string1="   ";

print string1.isspace();

string2="WELCOME TO WORLD OF PYT"

print string2.isspace();

Output:

```
>>>
True
False
```

4) len(string)

```
string1="    ";
print len(string1);
string2="WELCOME TO SSSIT"
print len(string2);
Output:
>>>
4
16
```

5) endswith(string)

```
string1="Welcome to SSSIT";
substring1="SSSIT";
substring2="to";
substring3="of";
print string1.endswith(substring1);
print string1.endswith(substring2,2,16);
print string1.endswith(substring3,2,19);
print string1.endswith(substring3);
```

Output:

```
>>>
True
False
False
False
>>>
```

6) find(string)

```
str="Welcome to SSSIT";
substr1="come";
substr2="to";
print str.find(substr1);
print str.find(substr2);
print str.find(substr1,3,10);
print str.find(substr2,19);
```

```
Output:
>>>
3
8
3
-1
>>>
```

7) index(string)

```
str="Welcome to world of SSSIT";

substr1="come";

substr2="of";

print str.index(substr1);

print str.index(substr2);

print str.index(substr1,3,10);

print str.index(substr2,19);
```

Output:

```
>>>
3
17
3
Traceback (most recent call last):
  File "C:/Python27/fin.py", line 7, in
    print str.index(substr2,19);
ValueError: substring not found
>>>
```

8) isalnum()

```
str="Welcome to sssit";
    print str.isalnum();
str1="Python47";
print str1.isalnum();
Output:
>>>
False
True
>>>
```

9) isalpha()

string1="HelloPython";

Even space is not allowed

print string1.isalpha();

string2="This is Python2.7.4"

print string2.isalpha();

Output:

```
>>>
True
False
>>>
```

10) isdigit()

```
string1="HelloPython";
print string1.isdigit();
string2="98564738"
print string2.isdigit();
Output:
>>>
False
True
```

11) islower()

string1="Hello Python";

print string1.islower();

string2="welcome to "

print string2.islower();

Output:

```
>>>
False
True
>>>
```

12) isupper()

```
string1="Hello Python";
print string1.isupper();
string2="WELCOME TO"
print string2.isupper();
Output:
>>>
False
True
>>>
```

13) lower()

```
string1="Hello Python";
print string1.lower();
string2="WELCOME TO SSSIT"
print string2.lower();
```

Output:

```
>>>
hello python
welcome to sssit
>>>
```

14) upper()

```
string1="Hello Python";
print string1.upper();
string2="welcome to SSSIT"
print string2.upper();
Output:
>>>
HELLO PYTHON
WELCOME TO SSSIT
>>>
```

15) startswith(string)

```
string1="Hello Python";
print string1.startswith('Hello');
string2="welcome to SSSIT"
print string2.startswith('come',3,7);
```

Output:

```
>>>
True
True
>>>
```

16) swapcase()

```
string1="Hello Python";
print string1.swapcase();
string2="welcome to SSSIT"
print string2.swapcase();
Output:
>>>
hELLO pYTHON
WELCOME TO sssit
>>>
```

17) lstrip()

string1="   Hello Python";

print string1.lstrip();

string2="@@@@@@@@welcome to SSSIT"

print string2.lstrip('@');

Output:

```
>>>
Hello Python
welcome to SSSIT
>>>
```

18) rstrip()

```
string1="    Hello Python     ";
print string1.rstrip();
string2="@welcome to SSSIT!!!"
print string2.rstrip('!');
Output:
>>>
          Hello Python
@welcome to SSSIT
>>>
```