PYTHON STRINGS

Strings are the simplest and easy to use in Python.

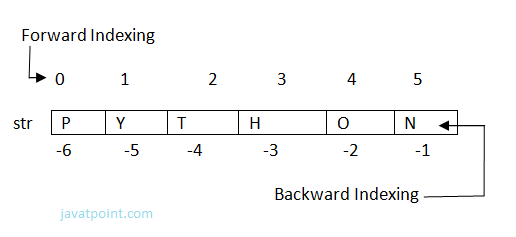
String pythons are immutable.

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:**



str[0]='P'=str[-6] , str[1]='Y' = str[-5]  ,  str[2] = 'T' = str[-4]  ,  str[3] = 'H' = str[-3]

str[4] = 'O' = str[-2]  ,  str[5] = 'N' = str[-1].

Simple program to retrieve String in reverse as well as normal form.

name="Rajat"

length=len(name)

i=0

**for** n **in** range(-1,(-length-1),-1):

**print** name[i],"\t",name[n]

    i+=1

**Output:**

>>>

R t

a a

j j

a a

t R

>>>

Strings Operators

There are basically 3 types of Operators supported by String:

1. Basic Operators.
2. Membership Operators.
3. 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'

>>>

|  |  |
| --- | --- |
| **Expression** | **Output** |
| '10' + '20' | '1020' |
| "s" + "007" | 's007' |
| 'abcd123' + 'xyz4' | 'abcd123xyz4' |

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

**Eg:**

'abc' + 3

>>>

**output:**

Traceback (most recent call last):

File "", line 1, in

'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:**

1. >>> 5\*"Vimal"

**Output:**

'VimalVimalVimalVimalVimal'

|  |  |
| --- | --- |
| **Expression** | **Output** |
| "soono"\*2 | 'soonosoono' |
| 3\*'1' | '111' |
| '$'\*5 | '$$$$$' |

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.

Membership Operators

Membership Operators are already discussed in the Operators section. Let see with context of String.

**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

1. True
2. >>> str5 **in** str2
3. >>> st5 **in** str2
4. True
5. >>> str6 **in** str3
6. True
7. >>> str4 **not** **in** str1
8. False
9. >>> str1 **not** **in** str4
10. 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:**

1. >>> "RAJAT"=="RAJAT"
2. True
3. >>> "afsha">='Afsha'
4. True
5. >>> "Z"<>"z"
6. True

**Explanation:**

The ASCII value of a is 97, b is 98, c is 99 and so on. The ASCII value of A is 65,B is 66,C is 67 and so on. The comparison between strings are done on the basis on ASCII value.

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:**

1. <string\_name>[startIndex:endIndex],
2. <string\_name>[:endIndex],
3. <string\_name>[startIndex:]

**Example:**

1. >>> str="Nikhil"
2. >>> str[0:6]
3. 'Nikhil'
4. >>> str[0:3]
5. 'Nik'
6. >>> str[2:5]
7. 'khi'
8. >>> str[:6]
9. 'Nikhil'
10. >>> str[3:]
11. 'hil'

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

String slice can also be used with Concatenation operator to get whole string.

**Eg:**

1. >>> str="Mahesh"
2. >>> str[:6]+str[6:]
3. 'Mahesh'

//here 6 is the length of the string.

String Functions and Methods:

There are many predefined or built in functions in String. They are as follows:

|  |  |
| --- | --- |
| 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. |

**Examples:**

**1) capitalize()**

1. >>> '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) 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:**

>>>

True2

False

False

False

>>>

**4) 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

>>>

**5) index(string)**

1. str="Welcome to world of SSSIT";
2. substr1="come";
3. substr2="of";
4. **print** str.index(substr1);
5. **print** str.index(substr2);
6. **print** str.index(substr1,3,10);
7. **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

>>>

**6) isalnum()**

1. str="Welcome to sssit";
2. **print** str.isalnum();
3. str1="Python47";
4. **print** str1.isalnum();

**Output:**

>>>

False

True

>>>

**7) isalpha()**

1. string1="HelloPython";    # Even space is not allowed
2. **print** string1.isalpha();
3. string2="This is Python2.7.4"
4. **print** string2.isalpha();

**Output:**

>>>

True

False

>>>

**8) isdigit()**

1. string1="HelloPython";
2. **print** string1.isdigit();
3. string2="98564738"
4. **print** string2.isdigit();

**Output:**

>>>

False

True

>>>

**9) islower()**

1. string1="Hello Python";
2. **print** string1.islower();
3. string2="welcome to "
4. **print** string2.islower();

**Output:**

>>>

False

True

>>>

**10) isupper()**

1. string1="Hello Python";
2. **print** string1.isupper();
3. string2="WELCOME TO"
4. **print** string2.isupper();

**Output:**

>>>

False

True

>>>

**11) isspace()**

string1="    ";

**print** string1.isspace();

string2="WELCOME  TO  WORLD OF PYT"

**print** string2.isspace();

**Output:**

>>>

True

False

>>>

**12) len(string)**

1. string1="    ";
2. **print** len(string1);
3. string2="WELCOME TO SSSIT"
4. **print** len(string2);

**Output:**

>>>

4

16

>>>

**13) lower()**

1. string1="Hello Python";
2. **print** string1.lower();
3. string2="WELCOME TO SSSIT"
4. **print** string2.lower();

**Output:**

>>>

hello python

welcome to sssit

>>>

**14) upper()**

1. string1="Hello Python";
2. **print** string1.upper();
3. string2="welcome to SSSIT"
4. **print** string2.upper();

**Output:**

>>>

HELLO PYTHON

WELCOME TO SSSIT

>>>

**15) startswith(string)**

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

**Output:**

>>>

True

True

>>>

**16) swapcase()**

1. string1="Hello Python";
2. **print** string1.swapcase();
3. string2="welcome to SSSIT"
4. **print** string2.swapcase();

**Output:**

>>>

hELLO pYTHON

WELCOME TO sssit

>>>

**17) lstrip()**

1. string1="    Hello Python";
2. **print** string1.lstrip();
3. string2="@@@@@@@@welcome to SSSIT"
4. **print** string2.lstrip('@');

**Output:**

>>>

Hello Python

welcome to SSSIT

>>>

**18) rstrip()**

1. string1="    Hello Python     ";
2. **print** string1.rstrip();
3. string2="@welcome to SSSIT!!!"
4. **print** string2.rstrip('!');

**Output:**

>>>

Hello Python

@welcome to SSSIT

>>>