Monday, 15 August 2022

* In python, Strings are the arrays of bytes representing Unicode characters.

* String can be created using single, double, or triple quotes.

* Creating String :

* With single quotes.

```
string = 'Happy Thoughts'
```

* With double quotes

```
string = "Happy Thoughts"
```

Single and double quotes are treated as same.

* With triple quotes

```
string = """ Hello ..!!
             Happy
                 Thoughts """
```

Multiline strings are created using triple quotes.

* String concatenation :-

* String concatenation is the operation of joining character strings end-to-end.
* In python, strings can be created concatenated using '+' operator.

For example :-

```
string1 = "Happy"
string2 = "Thoughts"

string3 = string1 + " " + string2
print(string3)
```

Output :

```
Happy Thoughts
```

* String Modification functions.

1) capitalize() :-

Capitalizes first letter of string.

Syntax :

```
String.capitalize()
```

2) center () :-

Returns a space-padded string with original string centered to a total of width columns.

Syntax :

```
String.center(width, fillchar)
```

3) count () :-

Counts how many times str occurs in string or in a substring of string if starting index beg and ending index are given.

Syntax :

```
string.count(sub, start=0, end=len(string))
```

sub : This is the substring to be searched

start : Search starts from this index.

end : Search ends at this index.

4) len() :-

Returns the length of the string.

Syntax :

```
len(string)
```

5) lower() :-

coverts all uppercase letters in string to lowercase.

Syntax :

```
string.lower()
```

6) replace() :-

Replaces all occurrences of old in string with new or at most max occurrences if max is given.

Syntax :

```
string.replace(old, new[, max])
```

7) split () :-

Split string according to delimeter str and returns list of substrings; split into at most num substrings if given.

Syntax :

```
string.split (str=" ", num=string.count(str))
```

8) strip() :-

Performs the removal of extra spaces in the string.

Syntax :

```
string.strip()
```

Operators :

Operators are used to perform operations on variables and values.

Arithematic operators:

+	addition
-	Subtraction
*	multiplication
/	Division
%	Modulus
**	exponentiation.
//	floor division

Python assignment operators:

=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x*= 5	x = x*5
/=	x /= 5	x = x /5
%=	x%= 5	x = x%5
//=	x//= 5	x = x//5
**=	x**= 5	x = x**5
&=	x&= 5	x = x &5
|=	x |= 5	x = x |5
^=	x^= 5	x = x ^5
>>=	x>>= 5	x = x >>5
<<=	x<<= 5	x = x <<5

Comparison operator:

==	equal
!=	not equal
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

Logical operators :

and	returns true when both statements are true
or	returns true when any of the condition is true
not	returns false if result is true & true if result is false

Identity operators.

is	returns if both variables are the same objects.
is not	returns true if both variables are not the same objects.

Membership operators:

in	returns true if a sequence with specified value is present in the object.
not in	returns true if a sequence with specified value is not present in the object.

Bitwise operator :

&	AND	sets each bit to 1 if both bits are 1.
|	OR	sets each bit to 1 if one of two bits is 1.
^	XOR	sets each bit to 1 if only one of two bits is 1.
~	NOT	Inverts all bits.
<<	zero fill left shift	Shift left by pushing zeros in from right and let the leftmost bits fall of.
>>	signed right shift	Shift right by pushing copies of the leftmost bit in from left, and let the rightmost bits fall of.