31 MOST USED APEX STRING METHODS

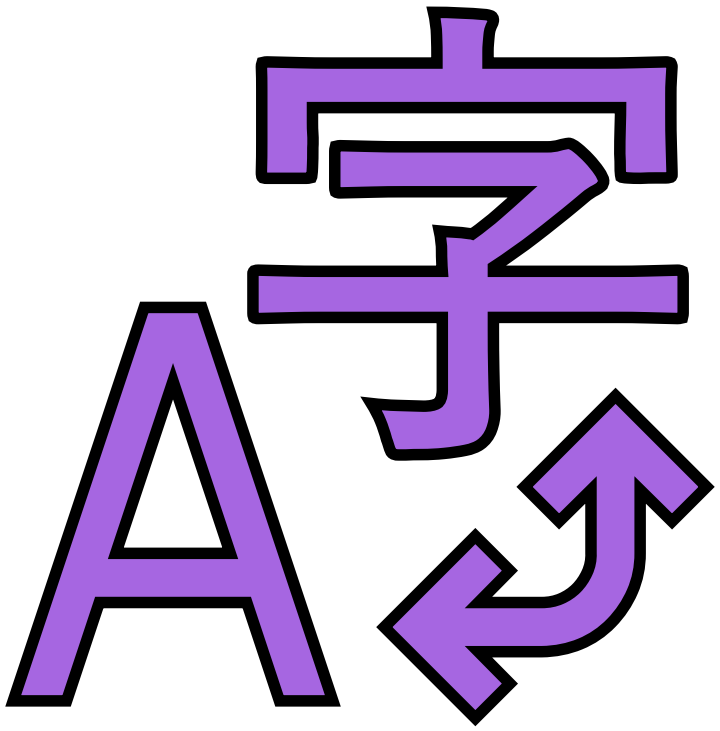
SFDC Lessons

Arun Kumar

toLowerCase()

Converts all characters in the String to lowercase using the default (English US) locale rules.

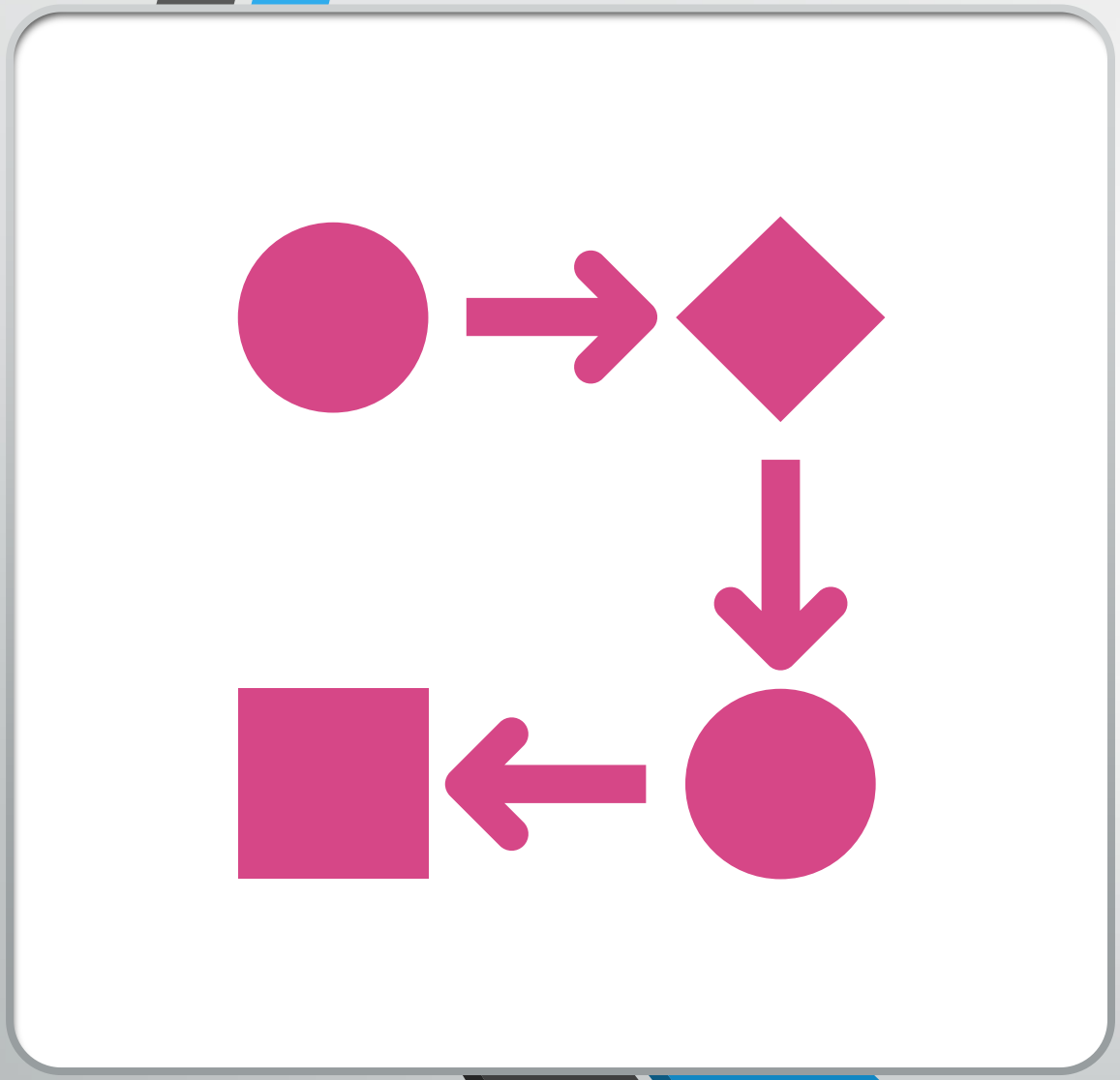
```
String s1 = 'ThIs iS ToLoWeRCaSE() MEthod ExAMple';

System.debug('>> '+ s1.toLowerCase());

>> this is tolowercase() method example
```



toUpperCase()

Converts all of the characters in the String to uppercase using the rules of the default (English US) locale.

```
String s1 ='this is a string';

System.debug('>> '+ s1.toLowerCase());

>> THIS IS A STRING
```



capitalize()

Returns the current String with the first letter changed to title case.

```
String s = 'hello trailblazer community';

System.debug('>> ' + s.capitalize() );

>> Hello trailblazer community
```

contains(substring)

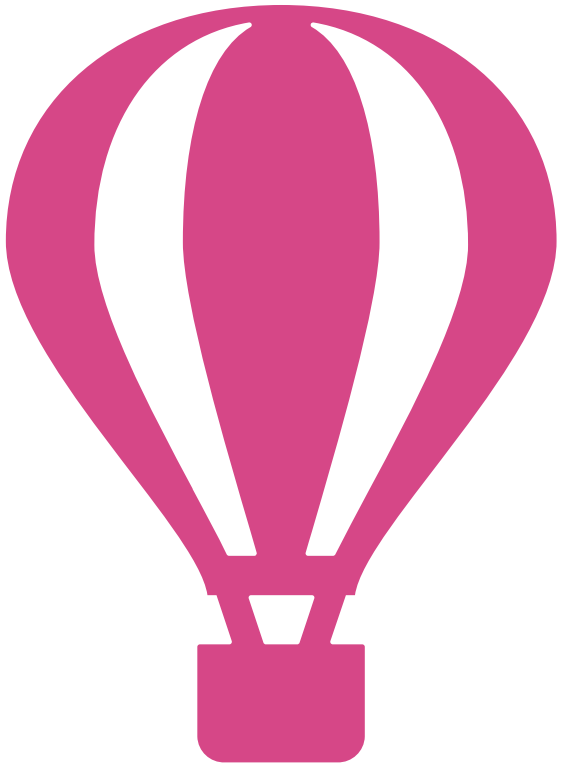
Returns true if and only if the String that called the method contains the specified sequence of characters in substring.

```
String str1 = 'Salesforce';
String str2 = 'force';
System.debug('>> ' + str1.contains(str2));

>> true
```

containsIgnoreCase (substring)

Returns true if the current String contains any of the characters in the specified String; otherwise, returns false

```
String s = 'welcome';
System.debug('>> ' + s.containsIgnoreCase('WEL');

>> true
```



containsAny(inputString)

Returns true if the current String contains any of the characters in the specified String; otherwise, returns false

```
String s = 'Salesforce';
Boolean b1 = s.containsAny('sx');
Boolean b2 = s.containsAny('x');
System.debug('>> '+ b1);
System.debug('>> '+ b2);

>> true
>> false
```

containsNone(inputString)

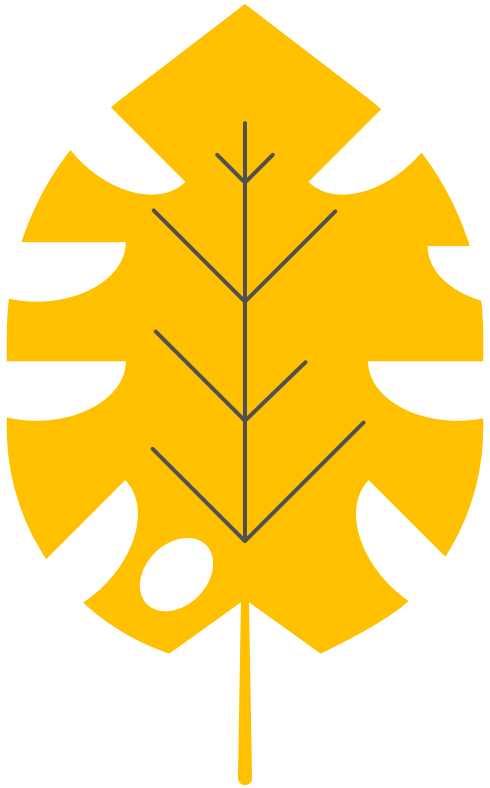
Returns true if the current String doesn't contain any of the characters in the specified String; otherwise, returns false.

```
String s1 = 'abcd';
System.debug('>> ' + s1.containsNone('fg'));

>> true
```

containsOnly(inputString)

Returns true if the current String contains characters only from the specified sequence of characters and not any other characters; otherwise, returns false.

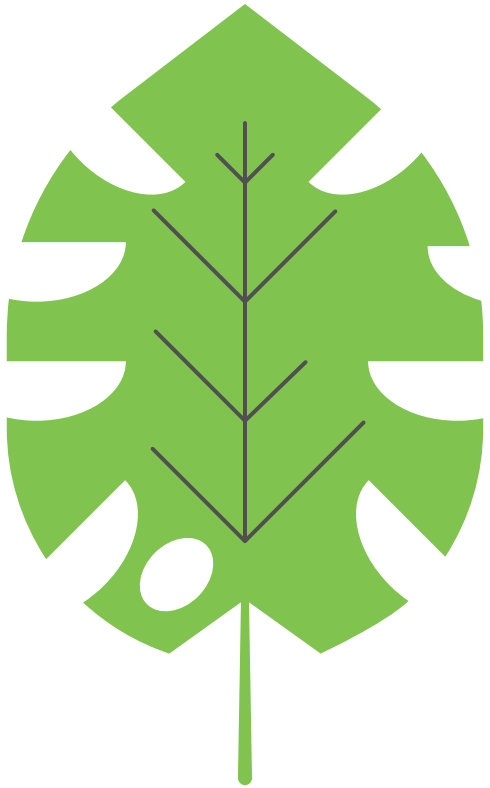
```
String s1 = 'salesforce';
String s2 = 'salesforce ohana';
system.debug('>> ' +
s1.containsOnly('salesofrce'));
system.debug('>> ' +
s2.containsOnly('salesforce'));

>> true
>> false
```



startsWith(prefix)

Returns true if the String that called the method begins with the specified prefix.

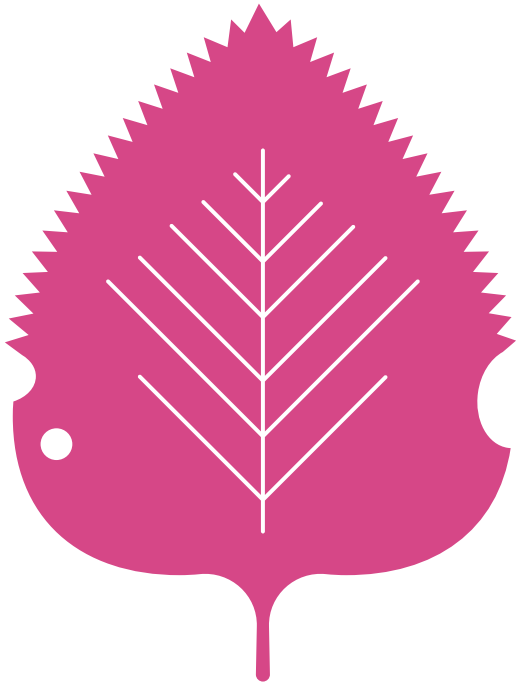
```
String s1 = 'AE86 HEC643 EK9';
System.debug('>> ' + s1.startsWith('AE86'));

>> true
```



startsWithIgnoreCase (prefix)

Returns true if the current String begins with the specified prefix regardless of the prefix case.

```
String s1 = 'AE86 vs EK9';
System.debug('>> ' +
s1.startsWithIgnoreCase('ae86'));

>> true
```



endsWith(suffix)

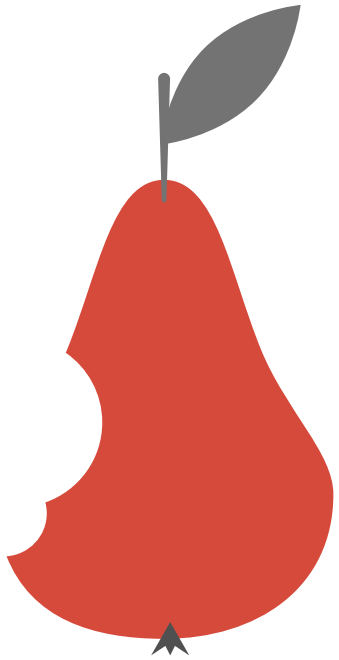
Returns true if the String that called the method ends with the specified suffix.

```
String s = 'Hello Trailblazers';
System.debug('>> ' + s.endsWith('Trailblazers'));

>> true
```

endsWithIgnoreCase (suffix)

Returns true if the String that called the method ends with the specified suffix.

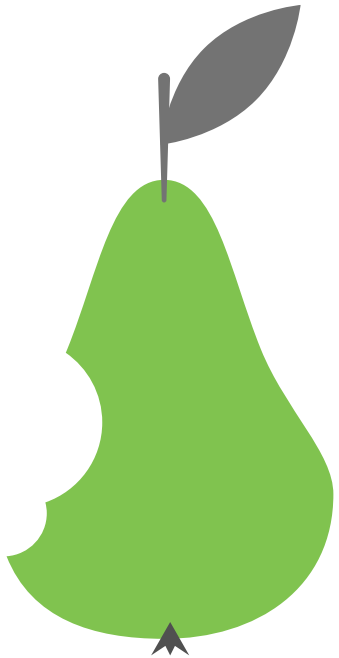
```
String s = 'Hello Trailblazers';
System.debug('>> ' +
s.endsWithIgnoreCase('trailblazers'));

>> true
```



substring(startIndex)

Returns a new String that begins with the character at the specified zero-based startIndex and extends to the end of the String.

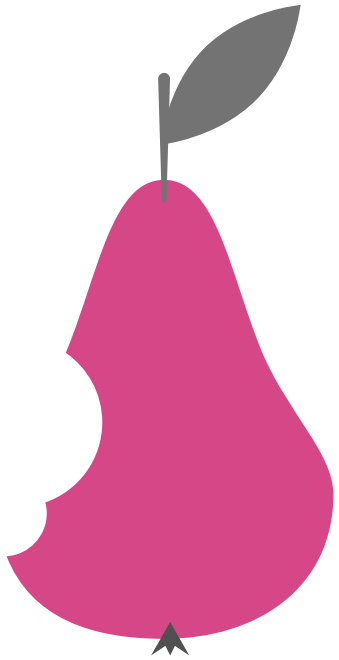
```
String s1 = 'salesforce';
System.debug('>> '+ s1.substring(5));

>> force
```



substring (startIndex, endIndex)

Returns a new String that begins with the character at the specified zero-based startIndex and extends to the character at endIndex - 1.

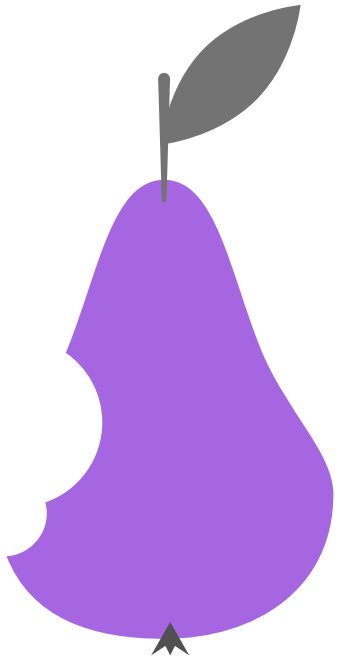
```
String str1 = 'Mulesoft'.substring(4, 8);
String str2 = 'Trailblazers'.substring(0, 5);

System.debug('>> ' + str1);
System.debug('>> ' + str2);

>> soft
>> Trail
```



substringAfter(separator)

Returns the substring that occurs after the first occurrence of the specified separator.

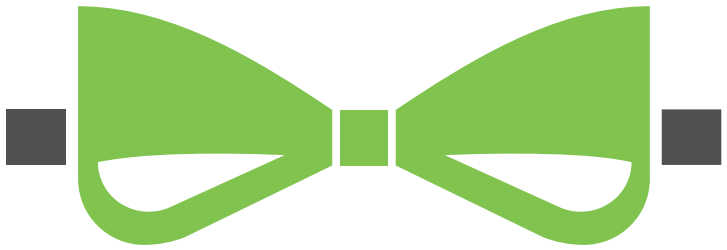
```
String s1 = 'Salesforce@2022';
System.debug('>> '+ s1.substringAfter('@') );

>> 2022
```



substringBefore(separator)

Returns the substring that occurs before the first occurrence of the specified separator.

```
String s1 = 'Salesforce@2022';
System.debug('>> '+ s1.substringBefore('@') );

>> Salesforce
```



trim()

Returns a copy of the string that no longer contains any leading or trailing white space characters

```
String s1 = '   Hello!   ';
String trimmed = s1.trim();
System.assertEquals('Hello!', trimmed);
System.debug('>> '+ trimmed );

>> Hello!
```

equals(stringOrId)

Returns true if the passed-in object is not null and represents the same binary sequence of characters as the current string. Use this method to compare a string to an object that represents a string or an ID.

```
// Compare a string to an object containing a
string
Object obj1 = 'force';
String str = 'force';
Boolean result1 = str.equals(obj1);
System.debug('>> ' + result1);

>> true
```

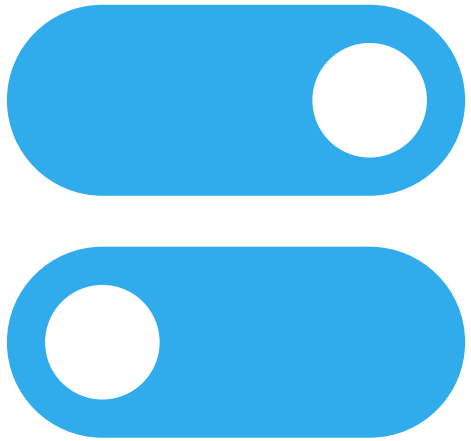
```
// 15-character ID
Id idValue15 = '001D000000Ju1zH';
// 15-character ID string value
String stringValue15 = '001D000000Ju1zH';
Boolean result2 = stringValue15.equals(IdValue15);
System.debug('>> ' + result2);

>> true
```

```
//  15-character ID and 18-character ID
Id idValue18 = '001D000000Ju1zHIAR';
Boolean result3 = stringValue15.equals(IdValue18);
System.debug('>> 1' +  result3);

>> true
```

equalsIgnoreCase (secondString)

Returns true if the passed-in object is not null and represents the same binary sequence of characters as the current string. Use this method to compare a string to an object that represents a string or an ID.

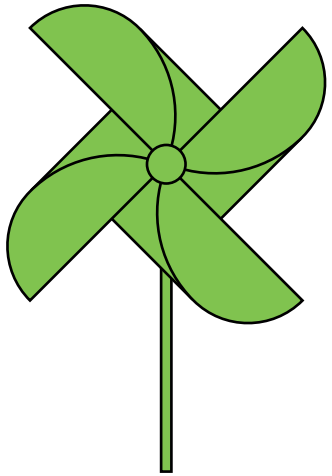
```
String myString1 = 'abcd';
String myString2 = 'ABCD';
Boolean result =
myString1.equalsIgnoreCase(myString2);
System.debug('>> '+ result);

>> true
```



swapCase()

Swaps the case of all characters and returns the resulting String by using the default (English US) locale.

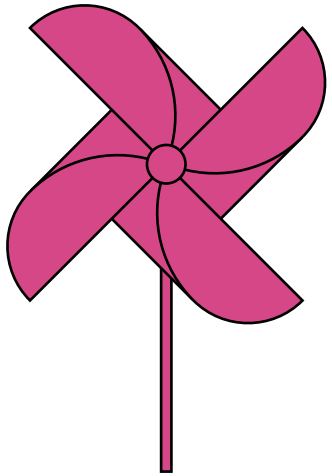
```
String s1 = 'salesFoRce.com';
String s2 = s1.swapCase();

System.debug('>> ' + s2 );

>> SALESfOrCE.COM
```



isAllLowerCase()

Returns true if all characters in the current String are lowercase; otherwise, returns false.

```
String allLowerString = 'trailblazers';
String mixedString = 'Trailblazers';
System.debug('>> '+
allLowerString.isAllLowerCase());
System.debug('>> '+ mixedString.isAllLowerCase());

>> true
>> false
```



isAllUpperCase()

Returns true if all characters in the current String are uppercase; otherwise, returns false.

```
String allUpperString = 'ABCDEFG';
String mixedString = 'Trailblazers';
System.debug('>> '+
allUpperString.isAllLowerCase());
System.debug('>> '+ mixedString.isAllLowerCase());

>> true
>> false
```

isAlpha()

Returns true if all characters in the current String are Unicode letters only; otherwise, returns false.

```
// Letters only
String s1 = 'dreamforce';
// Returns true
Boolean b1 = s1.isAlpha();
// Letters and numbers
String s2 = 'dreamforce 2022';
// Returns false
Boolean b2 = s2.isAlpha();
System.debug('>> ' +  b1);
System.debug('>> '+ b2);

>> true
>> false
```

isAlphaSpace()

Returns true if all characters in the current String are Unicode letters or spaces only; otherwise, returns false.

```
String alphaSpace = 'lightning component';
String notAlphaSpace = 'not found 404';

System.debug('>> '+ alphaSpace.isAlphaSpace());
System.debug('>> '+ notAlphaSpace.isAlphaSpace());

>> true
>> false
```

isAlphanumeric()

Returns true if all characters in the current String are Unicode letters or numbers only; otherwise, returns false.

```
String alphanumSpace = 'EUR86';
System.debug('>> '+
alphanumSpace.isAlphanumeric());

>> true
```

isAlphanumericSpace()

Returns true if all characters in the current String are Unicode letters, numbers, or spaces only; otherwise, returns false.

```
String alphanumSpace = 'AE 86';
System.debug('>> '+
alphanumSpace.isAlphanumericSpace());

>> true
```

isNumeric()

Returns true if the current String contains only Unicode digits; otherwise, returns false.

```
String numeric = '1234567890';
String alphanumeric = 'RG45';
String decimalPoint = '1.29';
System.debug('>> ' + numeric.isNumeric());
System.debug('>> ' + alphanumeric.isNumeric());
System.debug('>> ' + decimalpoint.isNumeric());

>> true
>> false
>> false
```

isNumericSpace()

isBlank(inputString)

Returns true if the specified String is white space, empty (''), or null; otherwise, returns false.

```
String blank = '';
String nullString = null;
String whitespace = '  ';
System.debug('>> '+ String.isBlank(blank));
System.debug('>> '+ String.isBlank(nullString));
System.debug('>> '+ String.isBlank(whitespace));

>> true
>> true
>> true
```

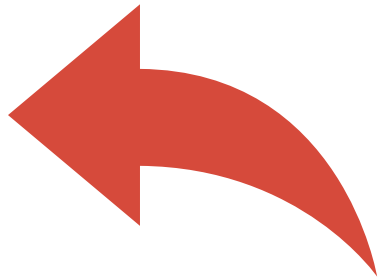
isNotBlank(inputString)

isEmpty(inputString)

Returns true if the specified String is empty (") or null; otherwise, returns false

```
String empty = '';
String nullString = null;
String whitespace = '  ';
System.debug('>> ' + String.isEmpty(empty));
System.debug('>> ' + String.isEmpty(nullString));
System.debug('>> ' + String.isEmpty(whitespace));

>> true
>> true
>> false
```

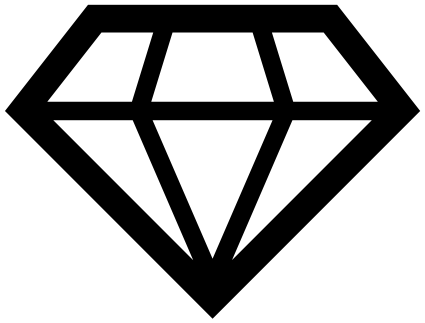
isNotEmpty(inputString)



reverse()

Returns a String with all the characters reversed.

```
String numeric = '1234567890';
String chars = 'ABCD';
System.debug('>> ' + numeric.reverse());
System.debug('>> ' + chars.reverse());

>> 0987654321
>> DCBA
```



valueOf(toConvert)

Returns a string representation of the specified object argument.

If the argument is not a String, the valueOf method converts it into a String by calling the toString method on the argument, if available, or any overridden toString method if the argument is a user-defined type. Otherwise, if no toString method is available, it returns a String representation of the argument.

```
List<Integer> ls = new List<Integer>{10,20};
String strList = String.valueOf(ls);
System.debug('>> '+ strList);

>> (10, 20)
```

valueOf(toConvert)

```
Integer myInteger = 22;
Decimal dec = 3.14159265;
System.debug('>> '+ String.valueOf(myInteger));
System.debug('>> '+ String.valueOf(dec));
>> 22
>> 3.14159265
```

```
//valueOf(datetimeToConvert)
DateTime dt = datetime.newInstance(1996, 6, 23);
String sDateTime = String.valueOf(dt);
System.assertEquals('1996-06-23 00:00:00',
sDateTime);
```

```
//valueOf(dateToConvert)
Date myDate = Date.Today();
String sDate = String.valueOf(myDate);

>> Returns a String that represents the specified
Date in the standard “yyyy-MM-dd” format.
```

For more info : [SFDC Lessons/apexStringMethods]

Thank You!