String Functions

For all the below examples, **str="COMPUTER"; Output** will be displayed as a **single line comment (//)**.

Quick Tip

1. **Function names start with lowercase and then the second word starts with uppercase letters.** Eg: **i**ndex**O**f();
2. **Topics asked in board questions are marked with** 

.length() (int)

This function is used to return the **length** of the string.

Syntax with example:

```
1. <int variable>=<string var>.length();
2. int Len=str.length();
3. //8
```

.charAt() (char)

This function returns the **character** from the given index.

Syntax with example:

```
1. <char variable>=<string var>.charAt(<index>);
2. char ch=str.charAt(2);
3. //O
```

.indexOf() (int)

This function returns the **index** of **first occurrence** of a character.

Syntax with example:

```
1. <int variable>=<string var>.indexOf(<character>);
2. int idx=str.indexOf('M');
3. //2
```

.indexOf(char ch, int start_index) (int)

This function returns the **index** of a given **character** from the given **index**.

Syntax with example:

```
1. <int var>=<String var>.indexOf(<char var>,<int var>);
2. char ch='M';
3. int ind=str.indexOf(ch, 1);
4. //2
```

.lastIndexOf(char ch) (int)

This function returns the **index** of the **last occurrence** of a given character.

Syntax with example:

```
1. <int var>=<String var>.lastIndexOf(char ch);
2. int ind=str.lastIndexOf('E');
3. //6
```

.substring(int start_index, int last_index) (String)

This function is used to extract a **set of characters** simultaneously from a given index upto the end of the String or till a given index.

Syntax with example:

```
1. <String var>=<String var>.substring(<int var>,<int var>);
2. String ext=str.substring(3);
3. //PUTER
```

.toLowerCase() (String)

This function is used to convert a given String to **lowercase** letters (entire string).

Syntax with example:

```
1. <String var>=<String var>.toLowerCase();
2. String lc=str.toLowerCase();
3. //computer
```

.toUpperCase() (String)

This function is used to convert a given String to **uppercase** letters (entire string).

Syntax with example:

```
1. <String var>=<String var>.toUpperCase();
2. String uc=str.toUpperCase(ind);
3. //COMPUTER
```

.replace(char old, char new) (String)

This function is used to **replace** a **character or a sequence of characters** in a String with a new character or sequence of characters. (**NOTE**: This does not work with int values)

Syntax with example:

```
1. <String var>=<String var>.replace(<char var>,<char var>);
2. String rep=str.replace("PUTER","PUTE");
3. //COMPUTE
```

.concat(String second) (String)

This function is used to **concatenate/join** two Strings together. (**NOTE**: This does not add any spaces in-between)

Syntax with example:

```
1. <String var>=<String var>.concat(s);
2. String s="STUDENT";
3. String con=str.(s);
4. //COMPUTERSTUDENT
```

.equals(String srt) (boolean)

This function is used to check for **equality** between two Strings. (**NOTE**: This function returns a **boolean** value. This function cannot be used for characters. //You can simply use == for characters. This can be used in if statements)

Syntax with example:

```
1. <boolean var>=<String var>.equals(<String var>);
2. String s=”COMPUTER”;
3. boolean chk=str.equals(s);
4. //true
```

. equalsIgnoreCase(String str) (boolean)

This function does the same function of .equals() function. The only difference is that it does not care about the case (It **ignores the case**).

Syntax with example:

```
1. <boolean var>=<String var>.equalsIgnoreCase(<String var>);
2. boolean chk=str.equalsIgnoreCase(“cOmPuTeR”); //true
```

.compareTo(String str) (int)

This function is used to **compare** two Strings. It also checks whether a String is **bigger or smaller** than the other and returns a suitable **int value**. It returns **0** if both are **equal**. A **positive** value when the **first is bigger** than the second and a **negative** value when the **second String is bigger** than the first. It returns the **no. of additional characters** when both the Strings' **first sequence of characters are equal** but the other has additional characters.

Syntax with example:

```
1. <int var>=<String var>.compareTo(<String var>);
2. String s=”SCIENCE”;
3. int cmp=str.compareTo(s);
4. //A, B, C, (C is the 3rd letter in the Alphabet and S is the 19th)
5. //the value of cmp will be-16 because (3-19=-16)
```

.compareToIgnoreCase(String str) (int)

This function does the same function as .compareTo but it **ignores the case**.

Syntax with example:

```
1. <int var>=<String var>.compareToIgnoreCase(<String var>);
2. int cmp=str.compareToIgnoreCase("cOmPuTeR");
3. //0
```

.trim() (String)

This function removes **spaces** at the **start and end** of the String. (**NOTE:** This function does not remove spaces in-between characters)

Syntax with example:

```
1. <String var>=<String var>.trim();
2. Str="    He llo World!  ";
3. String trm=str.trim();
4. //He llo World!
```

.startsWith(String str) (boolean)

This function is used to check if the given String is a **prefix** to the other.

Syntax with example:

```
1. <boolean var>=<String var>.startsWith(<String var>);
2. pfx="COM"
3. boolean chk=str.startsWith(pfx);
4. //true
```

.endsWith(String str) (boolean)

This function is used to check if a given String has a specified **suffix**.

Syntax with example:

```
1. <boolean var>=<String var>.ends with(<String var>);
2. boolean chk=str.endsWith("TER");
```