Kotlin String Operations

Kotlin String Operations

In this tutorial, we shall learn different string operations that are available in Kotlin programming language.

Initialize String

You can initialize a string variable or string constant using initialization operator `=`. The string literal should be enclosed in double quotes.

In the following example, we have initialized a string variable and a string constant.

```
/**
 * Kotlin - String Initialization
 */
fun main(args: Array<String&gt;) {
    //string constant
    val siteUrl = "https://www.tutorialkart.com/"

    //string variable
    var page = "Kotlin Tutorial"
}
```

You may reassign the var `page` string variable, but you cannot reassign any value for `siteUrl` after its first initialization.

Print String

To print a string, you can use builtin function print() or println().

In the following example, we have printed strings using println() and print() functions.

```
/**
 * Kotlin - Print string to console
 */
fun main(args: Array<String&gt;) {
    println("Hello World!")
    print("www.tutorialkart.com - ")
    print("Kotlin Tutorial")
}
```

Run the above Kotlin program, and you shall get the following output in your run console.

```
Hello World!
www.tutorialkart.com - Kotlin Tutorial
```

println() prints with a new line appended at the end of the string. print() does not print a new line at the end of the string. So, any print statements after print() shall be printed in the same line, like how the second and third strings printed in same line.

String Length

To get the length of a string, you can use length `attribute` on the string. Following example demonstrates how to get string length.

```
/**
 * Kotlin - String length
 */
fun main(args: Array<String&gt;) {
    var str = "Hello World!"
    println(str.length)
}
```

Run the above Kotlin program.

```

```

The number of characters in our string is 12. Hence the output 12.

More examples at Kotlin String Length.

String Equals

To check if two strings are equal, you can use equal to operator == or use String.equals() method.

In the following program, we shall use == to check if two strings are equal.

```
/**
 * Kotlin - String Equals - ==
 */
fun main(args: Array<String&gt;) {
    var string1 = "Hello World!"
    var string2 = "Helloworld!"
```

```
    var string3 = "Hello World!"

    //print the boolean values
    println(string1==string2)
    println(string1==string3)

    //use equal operator in conditional statement like if
    if(string1==string2){
        print("string1 and string2 are equal")
    }
    if(string1==string3){
        print("string1 and string3 are equal")
    }
}
```

Run the above Kotlin program.

```
false
true
string1 and string3 are equal
```

`==` operator returns true if both the strings are equal, else it returns false. You can use this as a boolean expression in conditional expressions.

In the following program, we shall use String.equals to check if two strings are equal.

```
/**
 * Kotlin - String Equals - String.equals()
 */
fun main(args: Array<String&gt;) {
    var string1 = "Hello World!"
    var string2 = "Helloworld!"
    var string3 = "Hello World!"

    //print the boolean values
    println(string1.equals(string2))
    println(string1.equals(string3))

    //use equal operator in conditional statement like if
    if(string1.equals(string2)){
        print("string1 and string2 are equal")
    }
    if(string1.equals(string3)){
        print("string1 and string3 are equal")
    }
}
```

Run the above Kotlin program.

```
false
true
string1 and string3 are equal
```

String.equals(otherString) function returns true if both the strings are equal, else it returns false. You can use this as a boolean expression in conditional expressions as shown in the above Kotlin example program.

String Capitalize

To capitalize the first character of the given string, you can use String.capitalize() method.

```
/**
 * Kotlin - Capitalize String
 */
fun main(args: Array<String&gt;) {
    var string1 = "hello world"
    var capitalized = string1.capitalize()
    print(capitalized)
}
```

Run the program, and String.capitalize() shall return a string with all the characters in this string transformed to uppercase characters.

```
Hello world
```

First character of the string has been capitalized.

More examples at Kotlin String Capitalize.

String endsWith

To check if a string ends with a specific string, use String.endsWith() method.

In the following example, we shall check if string `Hello World` ends with the string `World`.

```
/**
 * Kotlin - String.endsWith()
 */
fun main(args: Array<String&gt;) {
    var string1 = "Hello World"
    var string2 = "World"
    print(string1.endsWith(string2))
}
```

Run the program, and `string1.endsWith(string2)` returns true, because `string1` ends with `string2`.

```
true
```

String.endsWith(otherString) returns true if this string ends with the string passed as parameter. Else the method returns false.

String startsWith

Similar to that of above method, we can also check if a string starts with a specific string.

In the following example, we shall check if string `Hello World` starts with the string `Hello`.

```
/**
 * Kotlin - String.startsWith()
 */
fun main(args: Array<String&gt;) {
    var string1 = "Hello World"
    var string2 = "Hello"
    print(string1.startsWith(string2))
}
```

Run the program, and `string1.endsWith(string2)` returns true, because `string1` ends with `string2`.

```
true
```

Concatenate Strings

To concatenate two or more strings, you can use concatenation operator `+`.

In the following example, we shall concatenate two strings and assign it to `string1`. Then we shall concatenate four strings into a single string and assign it to `string2`.

```
/**
 * Kotlin - String Concatenation
 */
fun main(args: Array<String&gt;) {
    var string1 = "Hello World." + " Hello Again!"
    var string2 = "Welcome to " + "Kotlin Tutorial" + " by " + "TutorialKart."
    println(string1)
    println(string2)
}
```

Run the program, and `string1.endsWith(string2)` returns true, because `string1` ends with `string2`.

```
Hello World. Hello Again!
```

```
Welcome to Kotlin Tutorial by TutorialKart.
```

More examples at Kotlin String Concatenation.

Substring

To find substring of a string, you can use String.substring(int start, int end) method.

In the following example, we have taken a string, and found a substring of this string with start=4 and end=8.

```
/**
 * Kotlin - String.substring()
 */
fun main(args: Array<String&gt;) {
    var string1 = "www.tutorialkart.com"
    var subString = string1.substring(4, 8)
    println(subString)
}
```

The substring of this string, that starts at index 4 and ends at index 8 is `tuto`.

```
tuto
```

Split String

To split a string by delimiter, use String.split() method.

In the following example, we shall split a given string using delimiter `.`.

```
/**
 * Kotlin - String.split()
 */
fun main(args: Array<String&gt;) {
    var string1 = "www.tutorialkart.com"
    var chunks = string1.split(".")
    for (chunk in chunks) {
        println(chunk)
    }
}
```

Run the program, and you shall see the elements of string array printed. String.split(String delimiter) returns an array of Strings after splitting the string using delimiter.

```
www
tutorialkart
```

```
com
```

More detailed examples at Kotlin Split String.

Replace String

To replace a old value with a new value in a given string, use String.replace() method. replace() method takes two Strings are arguments: first one is old string value and the second one is new string value.

In the following example, we shall replace the old value `"World"` with a new value `"User"`, in the given string `"Hello World"`.

```
/**
 * Kotlin - String.replace()
 */
fun main(args: Array<String&gt;) {
    var string1 = "Hello World"
    var string2 = string1.replace("World", "User")
    println(string2)
}
```

Run the program, and String.replace() replaces “World” with “User” in the given string.

```
Hello User
```

More examples at Kotlin String Replace.

Conclusion

In this Kotlin Tutorial, we learned some of the String operations that are most frequently used in Kotlin application development.

Kotlin Java

- Kotlin Tutorial

Getting Started

- Setup Kotlin(Java) Project
- Kotlin Example Program
- Convert Java to Kotlin

- Kotlin Main Function
- Kotlin Loops
- Kotlin For Loop
- Kotlin While, Do While Loops
- Kotlin Repeat
- Kotlin Ranges
- Kotlin When

Object Oriented Concepts

Classes

- Kotlin - Class, Primary and Secondary Constructors
- Kotlin Sealed Class
- Kotlin Data Class
- Kotlin Enum
- Kotlin - Extension Functions

Inheritance

- Kotlin Inheritance
- Kotlin Override Method of Super Class

Abstraction

- Kotlin Abstraction
- Kotlin Abstract Class
- Kotlin - Interfaces
- Kotlin Null Safety

Exception Handling

- Kotlin Try Catch
- Kotlin Throw Exception
- Kotlin Custom Exception

Fix Compilation Errors

- Kotlin - Variable must be initialized
- Kotlin - Primary Constructor call expected

- Kotlin - Null can not be a value of a non-null type String
- Kotlin - Cannot create an instance of an abstract class

Kotlin - String Operations

- Kotlin - Compare Strings
- Kotlin - Replace String
- Kotlin - Split String
- Kotlin - Split String to Lines
- Kotlin - String Capitalize

Kotlin - Functions

- Kotlin Function - Default Arguments
- Kotlin - Use Function

Kotlin Collections

Kotlin List

- Kotlin List
- Kotlin List forEach

Kotlin File Operations

- Kotlin - Create File
- Kotlin - Read File
- Kotlin - Read File as List of Lines
- Kotlin - Write to File
- Kotlin - Append Text to File
- Kotlin - Check if File Exists
- Kotlin - Copy a File to Other
- Kotlin - Iterate through all files in a directory
- Kotlin - Delete Recursively
- Kotlin - Get File Extension

Kotlin Interview Q/A

- Kotlin Interview Questions

Kotlin Android

- Kotlin Android Tutorial

Useful Resources

- How to Learn Programming