Any value written within a pair of single quote or double quotes in R is treated as a string. Internally R stores every string within double quotes, even when you create them with single quote.

Rules Applied in String Construction

* The quotes at the beginning and end of a string should be both double quotes or both single quote. They can not be mixed.
* Double quotes can be inserted into a string starting and ending with single quote.
* Single quote can be inserted into a string starting and ending with double quotes.
* Double quotes can not be inserted into a string starting and ending with double quotes.
* Single quote can not be inserted into a string starting and ending with single quote.

Examples of Valid Strings

Following examples clarify the rules about creating a string in R.

a <- 'Start and end with single quote'

print(a)

b <- "Start and end with double quotes"

print(b)

c <- "single quote ' in between double quotes"

print(c)

d <- 'Double quotes " in between single quote'

print(d)

When the above code is run we get the following output −

[1] "Start and end with single quote"

[1] "Start and end with double quotes"

[1] "single quote ' in between double quote"

[1] "Double quote \" in between single quote"

Examples of Invalid Strings

e <- 'Mixed quotes"

print(e)

f <- 'Single quote ' inside single quote'

print(f)

g <- "Double quotes " inside double quotes"

print(g)

When we run the script it fails giving below results.

Error: unexpected symbol in:

"print(e)

f <- 'Single"

Execution halted

String Manipulation

Concatenating Strings - paste() function

Many strings in R are combined using the **paste()** function. It can take any number of arguments to be combined together.

Syntax

The basic syntax for paste function is −

paste(..., sep = " ", collapse = NULL)

Following is the description of the parameters used −

* **...** represents any number of arguments to be combined.
* **sep** represents any separator between the arguments. It is optional.
* **collapse** is used to eliminate the space in between two strings. But not the space within two words of one string.

Example

a <- "Hello"

b <- 'How'

c <- "are you? "

print(paste(a,b,c))

print(paste(a,b,c, sep = "-"))

print(paste(a,b,c, sep = "", collapse = ""))

When we execute the above code, it produces the following result −

[1] "Hello How are you? "

[1] "Hello-How-are you? "

[1] "HelloHoware you? "

Formatting numbers & strings - format() function

Numbers and strings can be formatted to a specific style using **format()** function.

Syntax

The basic syntax for format function is −

format(x, digits, nsmall, scientific, width, justify = c("left", "right", "centre", "none"))

Following is the description of the parameters used −

* **x** is the vector input.
* **digits** is the total number of digits displayed.
* **nsmall** is the minimum number of digits to the right of the decimal point.
* **scientific** is set to TRUE to display scientific notation.
* **width** indicates the minimum width to be displayed by padding blanks in the beginning.
* **justify** is the display of the string to left, right or center.

Example

# Total number of digits displayed. Last digit rounded off.

result <- format(23.123456789, digits = 9)

print(result)

# Display numbers in scientific notation.

result <- format(c(6, 13.14521), scientific = TRUE)

print(result)

# The minimum number of digits to the right of the decimal point.

result <- format(23.47, nsmall = 5)

print(result)

# Format treats everything as a string.

result <- format(6)

print(result)

# Numbers are padded with blank in the beginning for width.

result <- format(13.7, width = 6)

print(result)

# Left justify strings.

result <- format("Hello", width = 8, justify = "l")

print(result)

# Justfy string with center.

result <- format("Hello", width = 8, justify = "c")

print(result)

When we execute the above code, it produces the following result −

[1] "23.1234568"

[1] "6.000000e+00" "1.314521e+01"

[1] "23.47000"

[1] "6"

[1] " 13.7"

[1] "Hello "

[1] " Hello "

Counting number of characters in a string - nchar() function

This function counts the number of characters including spaces in a string.

Syntax

The basic syntax for nchar() function is −

nchar(x)

Following is the description of the parameters used −

* **x** is the vector input.

Example

result <- nchar("Count the number of characters")

print(result)

When we execute the above code, it produces the following result −

[1] 30

Changing the case - toupper() & tolower() functions

These functions change the case of characters of a string.

Syntax

The basic syntax for toupper() & tolower() function is −

toupper(x)

tolower(x)

Following is the description of the parameters used −

* **x** is the vector input.

Example

# Changing to Upper case.

result <- toupper("Changing To Upper")

print(result)

# Changing to lower case.

result <- tolower("Changing To Lower")

print(result)

When we execute the above code, it produces the following result −

[1] "CHANGING TO UPPER"

[1] "changing to lower"

Extracting parts of a string - substring() function

This function extracts parts of a String.

Syntax

The basic syntax for substring() function is −

substring(x,first,last)

Following is the description of the parameters used −

* **x** is the character vector input.
* **first** is the position of the first character to be extracted.
* **last** is the position of the last character to be extracted.

Example

# Extract characters from 5th to 7th position.

result <- substring("Extract", 5, 7)

print(result)

When we execute the above code, it produces the following result −

[1] "act"