# DAY 2: Strings

28 December 2024 11:55

# Day 2

## Strings

# DAY 2

## STRINGS

- Grouping with "" & {}
- Special characters as word
- How to add multi-line string
- string is, string length, string index
- string first, string last
- string match, string compare, string equal
- string map, trim
- string toupper/tolower
- regexp
- regsub
- Assignments

# String

- String consist of collection of arbitrary characters.
- Use to store text information.
- Single words do not require double quotes, while multiple words with spaces need to be in double quotes or curly braces to consider them as a single word. This is often referred as grouping.

## Character Type

# Character Type

- ASCII, Boolean, Upper, Lower, Integer, punct, space, wordchar, digit, xdigit

```
(TCL Scripting) 2 % string is boolean 1
1
(TCL Scripting) 3 % string is boolean type
0

(TCL Scripting) 5 % string is boolean true
1
(TCL Scripting) 6 % string is digit 123
1
(TCL Scripting) 7 % string is digit 12a
0
(TCL Scripting) 8 % string is ascii a
1
(TCL Scripting) 9 % string is ascii ae
1
(TCL Scripting) 10 % string is ascii =
1
(TCL Scripting) 11 % string is upper abc
0
(TCL Scripting) 12 % string is upper ABC
1
```

You Can Check Like This If String you Entered is Correct or Not.

# Grouping : “”, {}

- Substitution within double quotes can be performed normally, while substitution within curly braces may be blocked.
- Double quotes are used here to prevent spaces from being treated as delimiters.

```
(Downloads) 1 % set var1 12
12
(Downloads) 2 % puts Value of var1 is $var1
wrong # args: should be "puts ?-nonewline? ?channelId? string"
(Downloads) 3 % puts "Value of var1 is $var1"
Value of var1 is 12
(Downloads) 4 % puts {Value of var1 is $var1}
Value of var1 is $var1
```

```
(TCL Scripting) 14 % puts "Hello World"
Hello World
(TCL Scripting) 15 % puts Hello World
can not find channel named "Hello"
```

```
(TCL Scripting) 17 % puts {Hello World}
Hello World
```

Difference between "" and {}

```
(TCL Scripting) 18 % set var1 12
12
(TCL Scripting) 19 % set var2 14
14
(TCL Scripting) 20 % puts "The value of var1 $var1"
The value of var1 12
(TCL Scripting) 21 % puts {The value of var1 $var1}
The value of var1 $var1
(TCL Scripting) 22 %
```

Special characters as word : use "\"  write in between here  \""

```
(TCL Scripting) 23 % puts "\"How are you?\""
"How are you?"
(TCL Scripting) 24 %
```

```
(TCL Scripting) 23 % puts "\"How are you?\""
"How are you?"
(TCL Scripting) 24 % puts "or you can use this below meth
od"
or you can use this below method
(TCL Scripting) 25 % puts {How are you ?}
How are you ?
```

Multi-line strings

```
(TCL Scripting) 26 % set var1 "APB IS USE TO
> build
> Slow peripherals
> in Soc
>  "
APB IS USE TO
build
Slow peripherals
in Soc

(TCL Scripting) 27 %
```

```
(TCL Scripting) 27 % string is digit 123
1
(TCL Scripting) 28 % string is integer 456
1
(TCL Scripting) 29 % string is xdigit 456
1
(TCL Scripting) 30 % string is xdigit 1213a
1
(TCL Scripting) 31 %
```

# String length

- Calculate number of characters present in string
- Return decimal value equal to number of characters present in string including blank spaces

string length string_in

```
(TCL) 5 % string length "Hello World"
11

(TCL) 7 % set var1 [string length "Hello World"]
11
```

```
(TCL Scripting) 31 % string length Hello
5
(TCL Scripting) 32 % string length "Hello World"
11
(TCL Scripting) 33 % string length {Hello World}
11
(TCL Scripting) 34 %
```

```
(TCL Scripting) 34 % set var1 [string length "Ayush"]
5
```

# String index

- Return character at specific index provided by user or empty string if index is beyond range
- Index start at 0

string index string_in index_in

```
(TCL) 8 % set var1 "Hello"
(TCL) 9 % string index $var1 4
o
(TCL) 10 % string index $var1 2
l
(TCL) 11 % string index $var1 3
l
(TCL) 12 % string index $var1 0
H
```

| H | E | L | L | O |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |

```
(TCL Scripting) 36 % set var1 "hello
> "
hello

(TCL Scripting) 39 % string index hello 0
h
(TCL Scripting) 40 % string index hello 3
l
(TCL Scripting) 41 % string index hello 4
o
(TCL Scripting) 42 % string index $var1 4
o
(TCL Scripting) 43 % string index $var1 2
l
(TCL Scripting) 44 %
```

Use either that word or $var_name in which you have stored that word

```
(TCL Scripting) 44 % string index hello end
```

Will return last

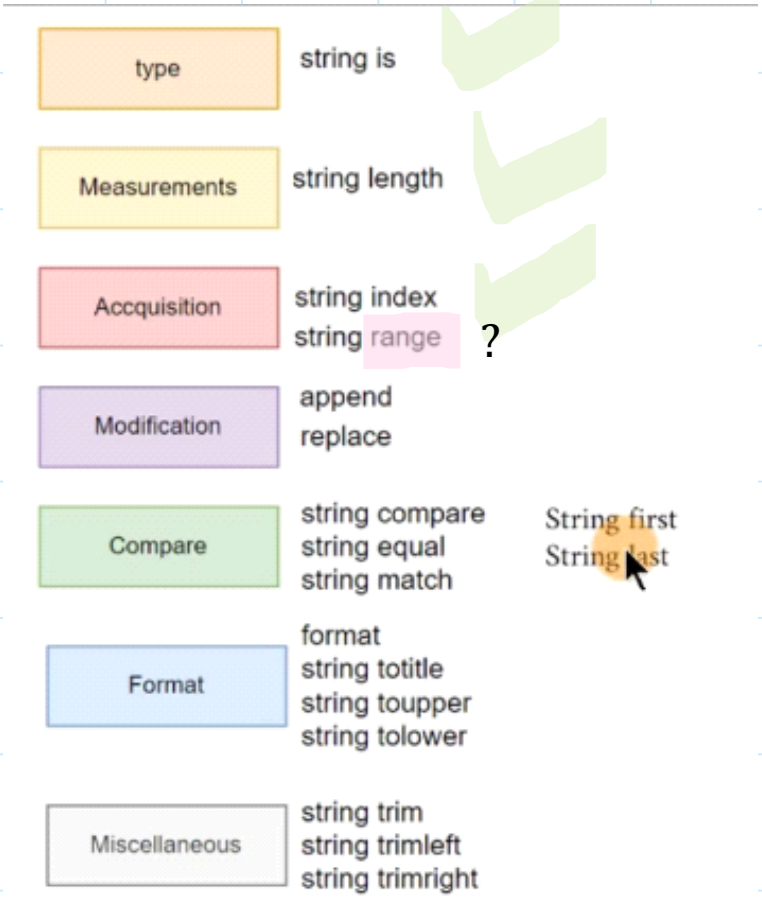
```
(TCL Scripting) 44 % string index hello end
o
(TCL Scripting) 45 % string index hello end-2
l
```

Will return last

Will return 2nd last

```
(TCL Scripting) 46 % string index hello 76
(TCL Scripting) 47 %
```

Using exceeding index
Returns empty




```
(TCL Scripting) 47 % set var1 Hello
Hello
(TCL Scripting) 48 % string first He $var1
0
```

Index 0 is retuned as
He is there at index 0
in Hello

```
(TCL Scripting) 49 % string first lo $var1
3
(TCL Scripting) 50 % string first o $var1
4
(TCL Scripting) 51 %
(TCL Scripting) 51 % string first e $var1
1
```

If something is not there
then -1 is we will get

```
(TCL Scripting) 52 % string first ex $var1
-1
(TCL Scripting) 53 % string first xyz' $var1
-1
(TCL Scripting) 54 % string first xyz $var1
-1
```

To check if particular
alphabet
is a part of the string

```
(TCL Scripting) 56 % string first l $var1 3
3
(TCL Scripting) 57 % string first l $var1 4
-1
(TCL Scripting) 58 %
```

## String first

- Return position where string is found in source string else return -1 is string is not found

# String first

- Return position where string is found in source string else return -1 is string is not found

string first string string_source <start_index>
string last string string_source <start_index>

```
(TCL) 16 % string first Hel $var1
0
(TCL) 17 % string first abc $var1
-1
(TCL) 18 % string first e $var1 2
-1
```

```
(TCL) 20 % set var1 HelloHello
HelloHello
(TCL) 21 % string first o $var1
4
(TCL) 22 % string last o $var1
9
```

String Last

C o n s o l e

Index: 0 1 2 3 4 5 6

```
(TCL Scripting) 59 % set var2 Console
Console
(TCL Scripting) 60 % string last o $var2
4
(TCL Scripting) 61 % string last n $var2
2
(TCL Scripting) 62 % string last e $var2
6
(TCL Scripting) 63 % string last s $var2
3
```

String Last

C o n s o l e

Index: 0 1 2 3 4 5 6

```
(TCL Scripting) 65 % string last o $var2
4
(TCL Scripting) 66 % string first o $var2
1
```

Difference between first and last will come

Difference between first and last will come when one string have
more than one repeated alphabet.

```
(TCL Scripting) 67 % set var3 banana
banana
(TCL Scripting) 68 % string first a $var3
1
(TCL Scripting) 69 % string last a $var3
5
```

Where is this a ?

String match

## String match

- String match command is used to determine if a given string matches a specified pattern.

```
string match ?-nocase? pattern string
```

**Parameters:**

- `-nocase` (optional): Makes the match case-insensitive.
- `pattern` : The pattern to match against. It can include special characters like `*` , `?` , and `[]` for more flexible matching.
- `string` : The string to be matched.

**Special Characters:**

- `*` : Matches any sequence of characters, including an empty string.
- `?` : Matches any single character.

wildcard

```
(TCL) 43 % string match test " test "
0
```

**Special Characters:**

- `*` : Matches any sequence of characters, including an empty string. wildcard
- `?` : Matches any single character.

```
(TCL) 43 % string match test " test "
0
```

```
(TCL) 45 % string match *test " test"
1
```
Character at start

```
(TCL) 46 % string match *test* " test "
1
```
Character at end

```
(TCL) 47 % string match te*ed "test failed"
1
```
Seqeuence of character in middle

```
(TCL) 48 % string match test* "test"
1
```
Empty character

```
(TCL Scripting) 70 % string match test "test"
1
(TCL Scripting) 71 % string match tast "test"
0
(TCL Scripting) 72 % string match -nocase TEst "test"
1
(TCL Scripting) 73 % string match TEst "test"
0
```

```
(TCL Scripting) 74 % string match *test "extratest"
1
(TCL Scripting) 75 % string match test "extratest"
0
(TCL Scripting) 78 % string match *t "extratestor"
0
```

```
(TCL Scripting) 79 % string match t?est "test"
0
(TCL Scripting) 80 % string match t?st "tast"
1
(TCL Scripting) 81 % string match t?st "test"
```

```
1
(TCL Scripting) 81 % string match t?st "test"
1
```

## String Compare

### String compare

- The string compare command compares two strings character by character, starting from the left side (i.e., the first character).

- **-1**: If `string1` is lexicographically less than `string2`.
- **0**: If `string1` is equal to `string2`.
- **1**: If `string1` is greater than `string2`.

```
(TCL) 25 % string compare a b
-1
(TCL) 26 % string compare d b
1
(TCL) 27 % string compare b b
0
```

"Lexicographically" refers to the order in which words or sequences are arranged based on the alphabetical order of their component letters.

```
(TCL Scripting) 82 % string compare a b
-1
(TCL Scripting) 83 % string compare b b
0
(TCL Scripting) 84 % string compare b a
1
```

$a < b \rightarrow -1$

$b = b \rightarrow 0$

$b > a \rightarrow 1$

```
(TCL Scripting) 85 % string compare abc def
-1
(TCL Scripting) 86 % string compare def abc
1
(TCL Scripting) 87 % string compare def def
0
```

Case-Sensitive Comparison:

- In a **case-sensitive** lexicographic order, where uppercase letters come before lowercase letters, "Ghi" would appear before "ghi".
  - The first character 'G' comes before 'g' in the ASCII character set.
  - Therefore, "Ghi" < "ghi". opposite as What it seems

```
(TCL) 32 % string compare Ghi ghi
-1
```

```
(TCL) 33 % string compare G6hi G7hi
-1
```

```
(TCL Scripting) 88 % string compare Def def
-1
(TCL Scripting) 89 % string compare Def DEf
1
```

# String equal

- In Tcl, you can use the string equal command to compare two strings character by character.

```
string equal ?-nocase? ?-length int? string1 string2
```

-nocase : This option makes the comparison case-insensitive.

- -length int: This option specifies the number of characters to compare. If the length is negative, it is ignored.
- The command returns 1 if the strings are identical and 0 if they are not

```
(TCL Scripting) 90 % string equal -nocase -length 5 Helloy helloy
1
(TCL Scripting) 91 % string equal -nocase -length 5 Helloy helloe
1
(TCL Scripting) 92 % string equal -nocase -length 5 Helloy hellue
0
(TCL Scripting) 93 % string equal -length 5 Helloy helloe
```

```
0
(TCL Scripting) 93 % string equal -length 5 Helloy helloe
0
```

# String map

- The string map command is used for replacing substrings within a string based on a mapping provided as a list of pairs.
- string map does not recheck previously replaced substrings in the same pass. It only looks for matches based on the current state of the string after all replacements from the mapping have been applied.
- Each substring is only replaced once based on the current state.

To replace h with l use this

```
(TCL Scripting) 3 % set var1 "hello"
hello
(TCL Scripting) 4 % string map {h l} $var1
lello
(TCL Scripting) 5 %
```

```
(TCL Scripting) 5 % puts $var1
hello
(TCL Scripting) 6 % put $var1
hello
(TCL Scripting) 7 % string map {"hello" "Ayush"} $
var1
Ayush
(TCL Scripting) 8 % puts $var1
hello
(TCL Scripting) 9 %
```

```
(TCL Scripting) 9 % set var2 "Hello~World~!"
Hello~World~!

(TCL Scripting) 11 % string map {"~" " " "!" " " } $var2
Hello World
(TCL Scripting) 12 %
```

replaced

## String trim / string trimleft / string trimright

- trim removes leading and trailing whitespaces.
- Trim left removes leading whitespaces.
- Trim right Removes trailing whitespaces.

```
(TCL) 2 % set var2 "Hello all "
Hello all
(TCL) 3 % string length $var2
10
(TCL) 4 % string trim $var2
Hello all
(TCL) 5 % set var2 [string trim $var2]
Hello all
(TCL) 6 % string length $var2
9
```

```
(TCL Scripting) 14 % set var1 " Hello World "
 Hello World

(TCL Scripting) 16 % string trim $var1
Hello World
```

```
(TCL Scripting) 16 % string trim $var1
Hello World
(TCL Scripting) 17 % string trimleft $var1
Hello World
(TCL Scripting) 18 % string trimright $var1
 Hello World
(TCL Scripting) 19 %
```

# String tolower/toupper

- tolower converts all characters in a string to lowercase. Each uppercase character (A-Z) is changed to its lowercase equivalent (a-z)
- toupper converts all characters in a string to uppercase. Each lowercase character (a-z) is changed to its uppercase equivalent (A-Z).
- Characters that are not alphabetic (such as punctuation or numbers) remain unchanged.

```
(TCL Scripting) 18 % string trimright $var1
Hello World

(TCL Scripting) 20 % string tolower $var1
hello world
(TCL Scripting) 21 %
```

```
(TCL Scripting) 18 % string trimright $var1
Hello World

(TCL Scripting) 20 % string tolower $var1
hello world
(TCL Scripting) 21 % string toupper $var1
HELLO WORLD
```