

MATLAB - STRINGS

http://www.tutorialspoint.com/matlab/matlab_strings.htm

Copyright © tutorialspoint.com

Creating a character string is quite simple in MATLAB. In fact, we have used it many times. For example, you type the following in the command prompt –

```
my_string = 'Tutorial's Point'
```

MATLAB will execute the above statement and return the following result –

```
my_string = Tutorial's Point
```

MATLAB considers all variables as arrays, and strings are considered as character arrays. Let us use the **whos** command to check the variable created above –

```
whos
```

MATLAB will execute the above statement and return the following result –

Name	Size	Bytes	Class	Attributes
my_string	1x16	32	char	

Interestingly, you can use numeric conversion functions like **uint8** or **uint16** to convert the characters in the string to their numeric codes. The **char** function converts the integer vector back to characters –

Example

Create a script file and type the following code into it –

```
my_string = 'Tutorial's Point';  
str_ascii = uint8(my_string)      % 8-bit ascii values  
str_back_to_char = char(str_ascii)  
str_16bit = uint16(my_string)    % 16-bit ascii values  
str_back_to_char = char(str_16bit)
```

When you run the file, it displays the following result –

```
str_ascii =  
    84  117  116  111  114  105  97  108  39  115  32  80  111  105  110  116  
str_back_to_char = Tutorial's Point  
str_16bit =  
    84  117  116  111  114  105  97  108  39  115  32  80  111  105  110  116  
str_back_to_char = Tutorial's Point
```

Rectangular Character Array

The strings we have discussed so far are one-dimensional character arrays; however, we need to store more than that. We need to store more dimensional textual data in our program. This is achieved by creating rectangular character arrays.

Simplest way of creating a rectangular character array is by concatenating two or more one-dimensional character arrays, either vertically or horizontally as required.

You can combine strings vertically in either of the following ways –

- Using the MATLAB concatenation operator **[]** and separating each row with a semicolon **;**. Please note that in this method each row must contain the same number of characters. For strings with different lengths, you should pad with space characters as needed.
- Using the **char** function. If the strings are of different lengths, char pads the shorter strings with trailing blanks so that each row has the same number of characters.

Example

Create a script file and type the following code into it –

```
doc_profile = ['Zara Ali           '; ...
              'Sr. Surgeon        '; ...
              'R N Tagore Cardiology Research Center'];
doc_profile = char('Zara Ali', 'Sr. Surgeon', ...
                  'RN Tagore Cardiology Research Center');
```

When you run the file, it displays the following result –

```
doc_profile =
Zara Ali
Sr. Surgeon
R N Tagore Cardiology Research Center
doc_profile =
Zara Ali
Sr. Surgeon
RN Tagore Cardiology Research Center
```

You can combine strings horizontally in either of the following ways –

- Using the MATLAB concatenation operator, **[]** and separating the input strings with a comma or a space. This method preserves any trailing spaces in the input arrays.
- Using the string concatenation function, **strcat**. This method removes trailing spaces in the inputs.

Example

Create a script file and type the following code into it –

```
name = 'Zara Ali           ';
position = 'Sr. Surgeon        ';
worksAt = 'R N Tagore Cardiology Research Center';
profile = [name ' ' position ' ' worksAt];
profile = strcat(name, ' ', position, ' ', worksAt);
```

When you run the file, it displays the following result –

```
profile = Zara Ali           , Sr. Surgeon
, R N Tagore Cardiology Research Center
profile = Zara Ali,Sr. Surgeon,R N Tagore Cardiology Research Center
```

Combining Strings into a Cell Array

From our previous discussion, it is clear that combining strings with different lengths could be a pain as all strings in the array has to be of the same length. We have used blank spaces at the end of strings to equalize their length.

However, a more efficient way to combine the strings is to convert the resulting array into a cell array.

MATLAB cell array can hold different sizes and types of data in an array. Cell arrays provide a more flexible way to store strings of varying length.

The **cellstr** function converts a character array into a cell array of strings.

Example

Create a script file and type the following code into it –

```
name = 'Zara Ali';
position = 'Sr. Surgeon';
worksAt = 'R N Tagore Cardiology Research Center';
profile = char(name, position, worksAt);
profile = cellstr(profile);
disp(profile)
```

When you run the file, it displays the following result –

```
{
 [1,1] = Zara Ali
 [2,1] = Sr. Surgeon
 [3,1] = R N Tagore Cardiology Research Center
}
```

String Functions in MATLAB

MATLAB provides numerous string functions creating, combining, parsing, comparing and manipulating strings.

Following table provides brief description of the string functions in MATLAB –

Function	Purpose
Functions for storing text in character arrays, combine character arrays, etc.	
blanks	Create string of blank characters
cellstr	Create cell array of strings from character array
char	Convert to character array <i>string</i>
iscellstr	Determine whether input is cell array of strings
ischar	Determine whether item is character array
sprintf	Format data into string
strcat	Concatenate strings horizontally
strjoin	Join strings in cell array into single string
Functions for identifying parts of strings, find and replace substrings	
ischar	Determine whether item is character array
isletter	Array elements that are alphabetic letters
isspace	Array elements that are space characters
isstrprop	Determine whether string is of specified category
sscanf	Read formatted data from string
strfind	Find one string within another
strrep	Find and replace substring
strsplit	Split string at specified delimiter

strtok	Selected parts of string
validatestring	Check validity of text string
symvar	Determine symbolic variables in expression
regexp	Match regular expression <i>casesensitive</i>
regexpi	Match regular expression <i>caseinsensitive</i>
regexprep	Replace string using regular expression
regextranslate	Translate string into regular expression

Functions for string comparison

strcmp	Compare strings <i>casesensitive</i>
strcmpi	Compare strings <i>caseinsensitive</i>
strncmp	Compare first n characters of strings <i>casesensitive</i>
strncmpi	Compare first n characters of strings <i>caseinsensitive</i>

Functions for changing string to upper- or lowercase, creating or removing white space

deblank	Strip trailing blanks from end of string
strtrim	Remove leading and trailing white space from string
lower	Convert string to lowercase
upper	Convert string to uppercase
strjust	Justify character array

Examples

The following examples illustrate some of the above-mentioned string functions –

Formatting Strings

Create a script file and type the following code into it –

```
A = pi*1000*ones(1,5);
sprintf(' %f \n %.2f \n %+.2f \n %12.2f \n %012.2f \n', A)
```

When you run the file, it displays the following result –

```
ans = 3141.592654
3141.59
+3141.59
3141.59
000003141.59
```

Joining Strings

Create a script file and type the following code into it –

```
%cell array of strings
str_array = {'red','blue','green','yellow','orange'};
```

```
% Join strings in cell array into single string
str1 = strjoin(str_array, "-")
str2 = strjoin(str_array, ",")
```

When you run the file, it displays the following result –

```
str1 = red-blue-green-yellow-orange
str2 = red,blue,green,yellow,orange
```

Finding and Replacing Strings

Create a script file and type the following code into it –

```
students = {'Zara Ali', 'Neha Bhatnagar', ...
            'Monica Malik', 'Madhu Gautam', ...
            'Madhu Sharma', 'Bhawna Sharma', ...
            'Nuha Ali', 'Reva Dutta', ...
            'Sunaina Ali', 'Sofia Kabir'};

% The strcmp function searches and replaces sub-string.
new_student = strcmp(students(8), 'Reva', 'Poulomi')
% Display first names
first_names = strtok(students)
```

When you run the file, it displays the following result –

```
new_student =
{
 [1,1] = Poulomi Dutta
}
first_names =
{
 [1,1] = Zara
 [1,2] = Neha
 [1,3] = Monica
 [1,4] = Madhu
 [1,5] = Madhu
 [1,6] = Bhawna
 [1,7] = Nuha
 [1,8] = Reva
 [1,9] = Sunaina
 [1,10] = Sofia
}
```

Comparing Strings

Create a script file and type the following code into it –

```
str1 = 'This is test'
str2 = 'This is text'
if (strcmp(str1, str2))
    printf('%s and %s are equal', str1, str2)
else
    printf('%s and %s are not equal', str1, str2)
end
```

When you run the file, it displays the following result –

```
str1 = This is test
str2 = This is text
ans = This is test and This is text are not equal
Loading [MathJax]/jax/output/HTML-CSS/jax.js
```