# String lists



## Overview

The **String list** **data type** is an ordered list of strings. This data type is returned, among others, by functions that return values of string fields in a selection of issues (**linked issues**, **sub-tasks**, and **subsets** ).

## Fixed values

A **string list** can also be written in literal form using the following format: **[*string*, *string*, ...]**.

> **Example**
>
> A number list with 5 elements: `["Blue", "Green", "Yellow", "Orange", "Red"]`

# String list functions

The following functions are intended to build expressions that return **string lists, strings** or **numbers**.

| Function | Input | Returned value |
|---|---|---|
| **filterByCardinality (string list** l, **comparison operator** operator, **number** n**)** | STRING [] NUMBER | Returns a STRING [] in **l** whose cardinality (i.e., the number of times it appears in list **l**) satisfies the comparison **cardinality operator n**. Available comparison operators: **=, !=, <, <=, >** and **>=** . Example: `filterByCardinality(["tiger", "tiger", "tiger", "tiger", "lion", "lion", "lion", "cat", "cat", "lynx"], <, 3)` returns `["cat", "lynx"]` . Example: `filterByCardinality(fieldValue(%{...components}, subtasks()), =, count(subtasks()))` returns a list with the **Components** present in all sub-tasks, i.e., those components common to all sub-tasks of current issue. |
| **filterByValue (string list** l, **comparison operator** operator, **string** s**)** | STRING [] STRING | Returns a STRING [] in **l** satisfying the comparison **string_in_list operator s**. Example: `filterByValue(["John", "Robert", "Kevin", "Mark"], ~, "r")` returns the list of string containing substring `"r"`. i.e., `["Robert", "Mark"]` |
| **filterByPredicate**(string list **l**, boolean expression **predicate** ) | STRING [] BOOLEAN | Returns a STRING [] in **l** that validate **predicate** . Argument **predicate** is a boolean expression, where `^%` is used for referencing string values in argument **l**. Example: `filterByPredicate(["book", "rose", "sword"], length(^%) > 4)` returns `["sword"]` . Example: `filterByPredicate(["book", "rose", "sword"], ^% in %{...summary} OR ^% in %{...description})` returns a list with those strings in first argument that also appear in issue **Summary** or **Description**. |

| | | |
|---|---|---|
| **append**(string list **l**, string list **m**) | STRING [] | Returns a STRING [] with all strings in arguments **l** and **m**. Duplicated string may appear in output. Use function **union**(l, m) instead, if you want to avoid repetitions.<br>Example: `append(["blue", "red", "green"], ["red", "green", "yellow"])` returns `["blue", "red", "green", "red", "green", "yellow"]` .<br><br>Example: `append(fieldValue(%{... fixVersions}, subtasks()), fieldValue(% {...fixVersions}, linkedIssues("is blocked by")))` returns a string list with **Fix Version /s** of sub-tasks and blocking issues. |
| **union**(string list **l**, string list **m**) | STRING [] | Returns a STRING [] with all strings in argument **l** or in argument **m** without duplicated strings.<br>Example: `union(["blue", "red", "green"], ["red", "green", "yellow"])` returns `["blue", "red", "green", "yellow"]` .<br>Example: `union(fieldValue(%{... fixVersions}, subtasks()), fieldValue(% {...fixVersions}, linkedIssues()))` returns the list of **Fix Version/s** selected among all sub-tasks and linked issues. |
| **except**(string list **l**, string list **m**) | STRING [] | Returns a STRING [] with all strings in argument **l** which are not in argument **m**. Duplicated strings in **l** may appear in output. Use function **distinct()** to remove them if you need to.<br>Example: `except(["blue", "red", "green", "black"], ["red", "green", "yellow"])` return s `["blue", "black"]` .<br>Example: `except(fieldValue(%{... fixVersions}, subtasks()), fieldValue(% {...fixVersions}, linkedIssues()))` returns the list of **Fix Version/s** in sub-tasks and not in linked issues. |
| **intersect**(string list **l**, string list **m**) | STRING [] | Returns a STRING [] with all strings in argument **l** and **m** simultaneously.<br>Example: `intersect(["blue", "red", "green", "black"], ["red", "green", "yellow"])` returns `["red", "green"]` .<br>Example: `union(fieldValue(%{... fixVersions}, subtasks()), fieldValue(% {...fixVersions}, linkedIssues()))` returns the list of **Fix Version/s** common to sub-tasks and linked issues. |
| **distinct**(string list **l**) | STRING [] | Returns a STRING [] with all strings in list **l** without any duplication.<br>Example: `distinct(["blue", "green", "yellow", "blue", "yellow"])` returns `["blue", "green", "yellow"]` .<br>Example: `distinct(fieldValue(%{... assignee}, subtasks()))` returns the list of assignees to sub-tasks, with only one occurrence per user, although a user may have more than one sub-task assigned. |
| **count**(string list **l**) | STRING [] | Returns the NUMBER of string values in **l**.<br>Example: `count(["blue", "red", "blue", "black"])` returns **4** .<br>Example: `count(distinct(fieldValue(%{... components}, subtasks())))` returns the number of **Components** selected among all sub-tasks. |
| **count**(string **s**, string list **l**) | STRING<br>STRING [] | Returns the NUMBER of times **s** appears in **l**.<br>Example: `count("blue", ["blue", "blue", "red", "red", "blue", "green"])` returns **3** . |

| | | |
|---|---|---|
| **first**(string **l**) | STRING [] | Returns the first element in STRING list **l**, or `null` if **l** is an empty list.<br>Example: **`first(["blue", "red", "green"])`** returns **`"blue"`** . |
| **last**(string **l**) | STRING [] | Returns the first element in STRING list **l**, or `null` if **l** is an empty list.<br>Example: **`last(["blue", "red", "green"])`** returns **`"green"`** . |
| **nthElement**(string list **l**, number **n**) | STRING []<br>NUMBER | Returns element at position **n** in STRING list **l** , where **n >= 1** and **n <= count(l)**. Returns `null` if **n** is greater than the number of elements in **l**.<br>Example: **`nthElement(["blue", "red", "green"], 2)`** returns **`"red"`** . |
| **getMatchingValue**(string **key**, string list **key_list**, string list **value_list**) | STRING<br>STRING [] | Returns STRING value in **value_list** that is in the same position as string **key** is in **key_list**, or in case **key** doesn't exist in **key_list** and **value_list** has more elements than **key_list**, the element of **value_list** in position **`count(key_list) + 1`** .<br>Example: **`getMatchingValue("Spain", ["USA", "UK", "France", "Spain", "Germany"], ["Washington", "London", "Paris", "Madrid", "Berlin"])`** returns **`"Madrid"`** . |
| **getMatchingValue**(string **key**, string list **key_list**, string list **value_list**) | STRING<br>STRING [] | Returns STRING value in **value_list** that is in the same position as numeric **key** is in **key_list**, or in case **key** doesn't exist in **key_list** and **value_list** has more elements than **key_list**, the element of **value_list** in position `count(key_list) + 1`.<br>Example: **`getMatchingValue(8, [2, 4, 6, 8, 10], ["Washington", "London", "Paris", "Madrid", "Berlin"])`** returns **`"Madrid"`** . |
| **sublist**(string list **l**, number **indexFrom**, number **indexTo**) | STRING []<br>NUMBER | Returns a STRING [] with elements in **l** from **indexFrom** index to **indexTo** index. Having **indexFrom >= 1** and **indexFrom <= count(l)** and **indexTo >= 1** and **indexTo <= count(l)** and **indexFrom <= indexTo**.<br>Example: **`sublist(["red", "green", "blue", "purple", "white"], 2, 4)`** returns **`["green", "blue", "purple"]`** . |
| **indexOf**(string **element**, string list **l**) | STRING<br>STRING [] | Returns the index NUMBER of string **element** in string list **l**. **Zero** is returned when **element** is not found in **l**.<br>Example: **`indexOf("blue", ["red", "blue", "green"])`** returns **`2`** . |
| **sort**(string list **l**, order) | STRING [] | Returns a STRING [] with elements in **l** lexicographically ordered. Available orders are **ASC** (for ascending order) and **DESC** (for descending order).<br>Example: **`sort(["red", "blue", "green"], ASC)`** returns **`["blue", "green", "red"]`** . |
| **textOnStringList**(string list **strings**, string **text_expression**) | STRING [] | Returns a STRING [] resulting of evaluating **text_expression** against each of the strings in argument **strings**. Argument **text_expression** is an expression that returns a string, where **^%** represents each string in argument **strings**.<br>Example: **`textOnStringList(["albert", "riCHard", "MARY"], capitalizeWordsFully(^%))`** returns **`["Albert", "Richard", "Mary"]`** . |
| **mathOnStringList**(string list **strings**, number **math_time_expression**) | STRING [] | Returns a NUMBER [] resulting of evaluating **math_time_expression** against each of the issues in argument **issues**. Argument **math_time_expression** is a math/time expression, where **^%** represents each string in argument **strings**.<br>Example: **`mathOnStringList(["a", "ab", "abc", "abcd", "abcde"], length(^%))`** returns **`[1, 2, 3, 4, 5]`** . |

## Examples

| Input | Output |
| --- | --- |
| `["red", "blue", "green"]` | A string list with the names of 3 colors |
| `fieldValue(%{...summary}, subtasks())` | Returns the list of s**ummaries of sub-tasks** of the current issue |
| `toStringList(%{...components})` | Returns a list with the **names of the components** of the current issue. |
| `distinct(toStringList(toString (fieldValue(%{...components}, subtasks())), ","))` | Returns a string list with all the components present in the sub-tasks of the current issue **without duplicates**. |