

SCRIPT REFERENCE

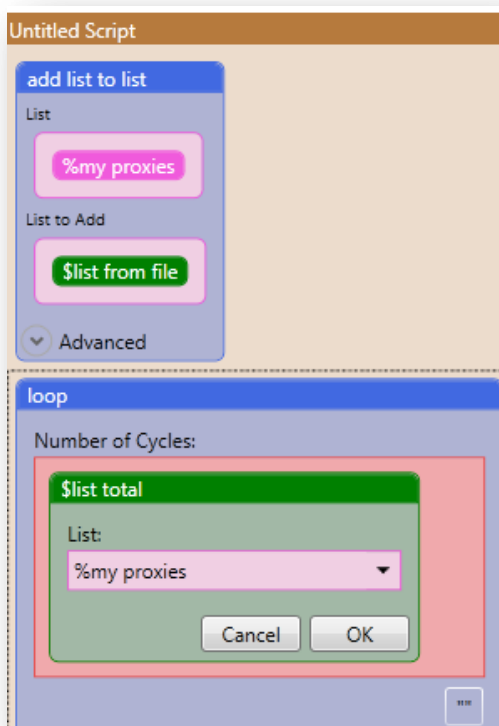
UBot Studio Version 4

The Variable and File Functions

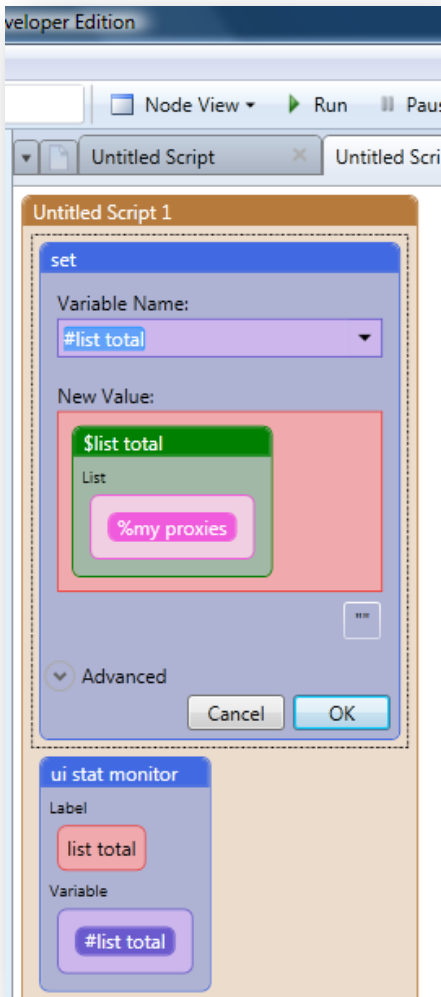
Variable Functions

List Total

This command will grab the total of all the items within a list. It is often used along with the loop command to cause a task to loop according to the number of items are within the loop.



In this example, we will set the list total of a list to a variable, and display the results on our UI using the UI stat monitor.



When we run our script, we are going to see the list total of our script displayed on our UI above the browser area.

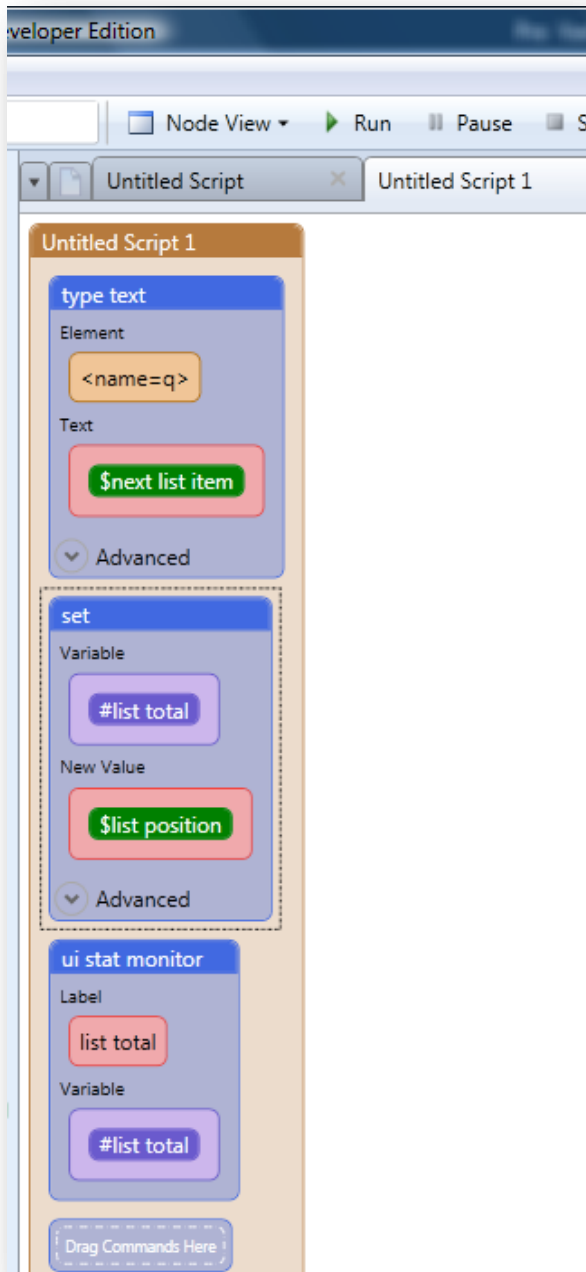


In all, the list total function contains the value for the total of the list you choose. You can use the value however you need to.

List Position

This command is similar to the list total command, except, instead of grabbing the total for an entire list, it grabs an item's position in a list according to what list item you used. This function works only with the next list item function and the previous list item functions.

In this example, we will display the list position of a next list item on the ui status monitor.



When we run our script, the list position of the list item we just used will be displayed on the UI.

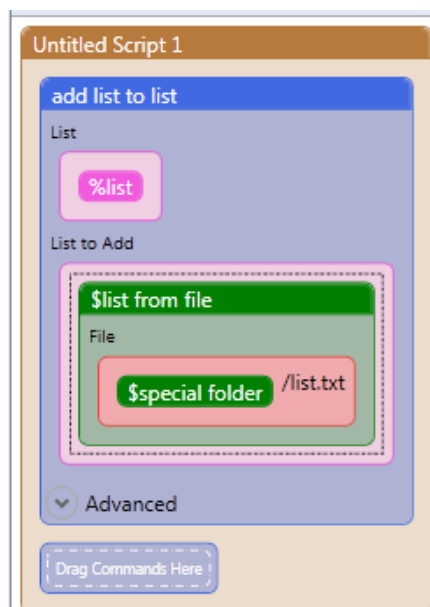
list total 2

Next List Item

This function will return the list item at the list's current position, and then increment the list's position by one.

This function is the easiest way to loop through a list sequentially (from position 0 to the end of the list).

Given a list containing the numbers 1–25, we will loop through the list using the \$next list item function:

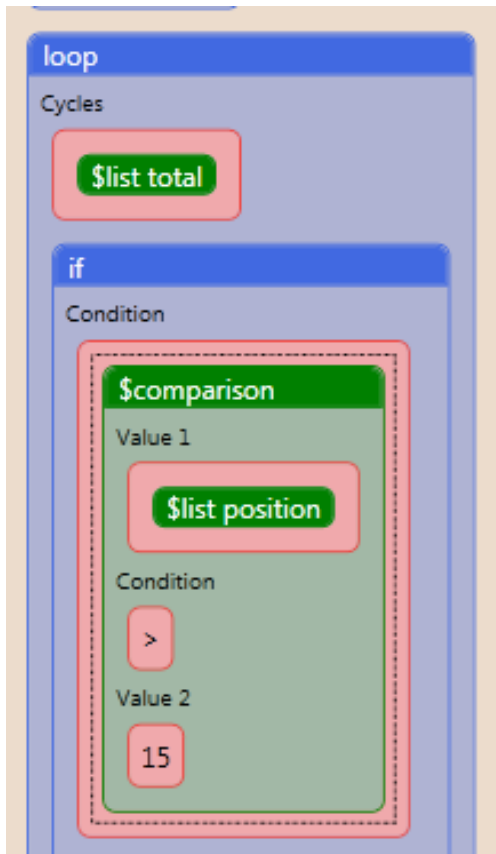


We will create the loop using the \$list total function:



Inside the loop we will use an if/then statement to set a condition.

In this case we will simply stop the script when the list item is greater than 15:



Place an if/then node in the scripting area and click on the qualifier place holder to highlight it. Drag a comparison qualifier (from the qualifier menu under Functions) into the if then else command.

For the first value, choose \$list position from the variables functions menu and place it in the area for the First value in the comparison qualifier. Select the greater than symbol from the drop down menu within the qualifier labeled "Condition" and type the number 15 into the area for the Second Value.

Proper Edition

Node View ▶ Run ||

Untitled Script × Untitled

loop

Cycles

\$list total

if

Condition

\$comparison

Value 1

\$list position

Condition

>

Value 2

15

then

stop script

Drag Commands Here

else

type text

Element

<name=q>

Text

\$next list item

Advanced

```
graph TD; Loop[loop] --> Cycles[Cycles]; Loop --> If[if]; Loop --> Then[then]; Loop --> Else[else]; If --> Comparison["$comparison"]; Comparison --> V1["Value 1"]; V1 --> ListPos["$list position"]; Comparison --> V2["Value 2"]; V2 --> 15["15"]; Comparison --> Cond[">"]; Then --> Stop["stop script"]; Else --> TypeText["type text"]; TypeText --> Element["<name=q>"]; TypeText --> Text["$next list item"];
```


Now drag a stop script command into the then command. Place a type text command into the else command to have it fill a field of your choice, as long as the condition we set is met.

When run, this script will loop through the list until it reaches a list position greater than 15 and stop the script.

It important to note that if you place a set list position node just above the loop node you can loop through the list as many times as you like without the worry of exceeding the range of the list. The set list position command effectively “resets” the list back to the first position.

Previous List Item

This function will return the list item at the list's current position, and then decrement the list's position by one.

This function works exactly the same as \$next list position except you would work from the end of a list and work up it.

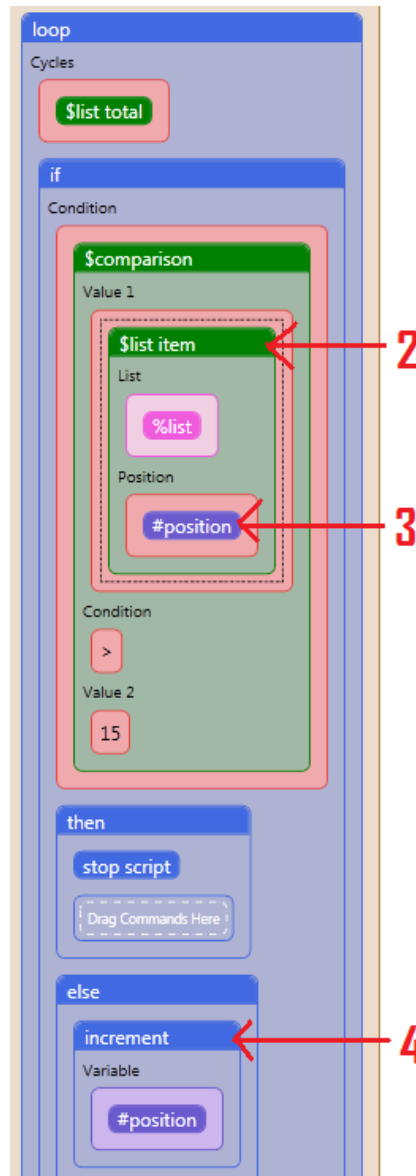
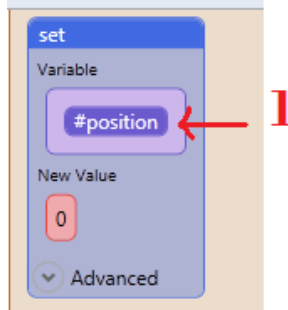
So instead of looping through a list from position 0 to position 9, you loop through it from position 9 to position 0.

This function is the easiest way to loop through a list sequentially (from the very last list position to the beginning of the list [position 0]).

List Item

This function will return the list item at the specified position. You can either specify the position at the time you choose the function, or you can set the position in a variable.

The following image shows the position being set in a variable for the purpose of looping through a list:



- 1) Set the variable called "position" to 0
- 2) Compare the list item using the #position variable as the position (3)

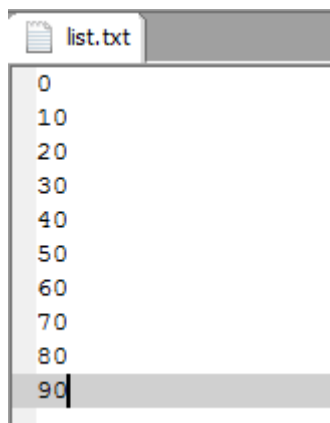
4) Increment the variable "position by 1

By incrementing the variable at the bottom of the loop it will change to 1 in the second cycle of the loop (2 in the third cycle, etc.). So the loop will compare the item in position 0, then position 1 all the way down to the end of the list, or in this case, under the condition returns true. (It finds a number greater than 15).

Read File

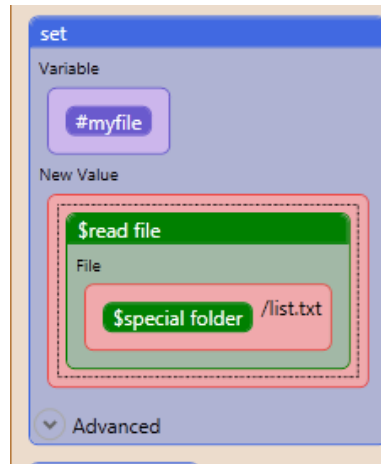
This function loads a file from a location on your computer. You can also call a file from a URL location by placing the url within the function.

This function is particularly useful when used in conjunction with the \$replace function. The reason is it loads the text as a single block of text that can be manipulated without looping through the lines of text. The list in this example contains the following numbers:



The idea is to replace all of the 0's with 9's. We will use the read file function to achieve this.

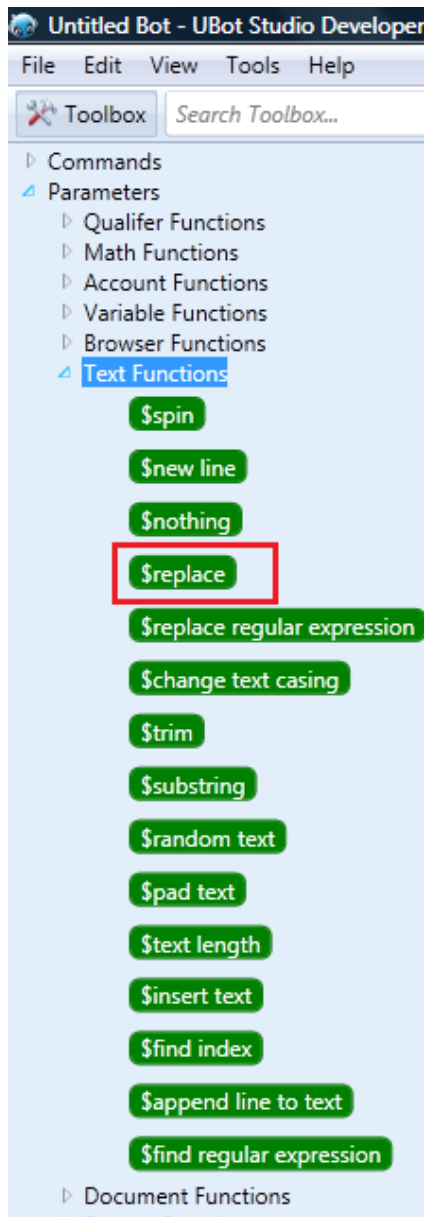
The first step is to set a variable. In this case we will call it #myfile. The content of the variable will be the \$read file function which will call the file you select. In this case my file is a list of numbers:



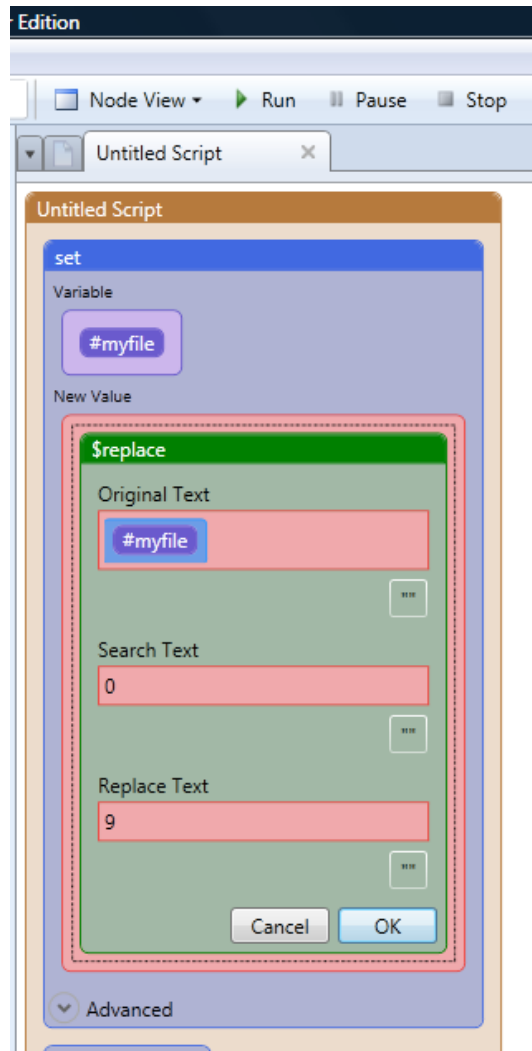
Ultimately we want the list to look like this:

```
9  
19  
29  
39  
49  
59  
69  
79  
89  
99
```

So the next step is to add another set command, but keep in mind we are setting the same variable again. In this case the set variable will be #myfile again, only this time the content will be a replace command. The reason this works is because the variable is already set to a block of text (not a line by line list):



Once you have chosen the replace function you will add the source (in this case is the original variable #myfile), the text to search for (in this case "0" and what to replace it with (in this case "9") :



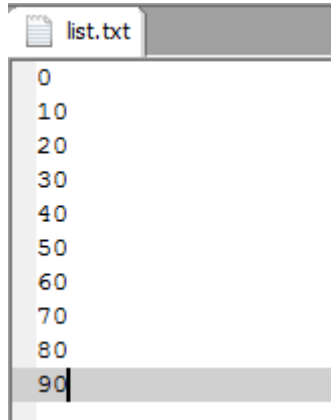
At this point just use a save to file command, and save the variable #myfile to the same file you read from. Your new file will have the list we wanted:

```
9
19
29
39
49
59
69
79
89
99
```

List From File

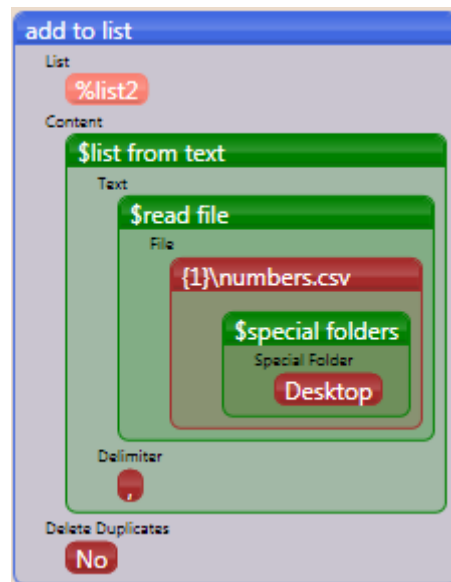
This function, when combined with the “add to list to list” command, pulls information from a file into a list. Items are separated with a new line.

Very simple, if you have a text file that contains a list of numbers, you can create a list in UBot using the content of the text file:



Simply drag the “add to list to list” command into the scripting area from the variable commands menu, and choose list from file:

You can now create a list from this file in UBot using the following code:



The first command is “add to list to list” from the variable commands menu. You will then choose a list (or create a new one by providing a new name), and use the \$list from text function as the content (from the variable functions menu):

In the list from text function, you will choose the \$read file function from the text functions list in the tool box. Browse for the file you wish to create a list from. You will also select the delimiter you wish to use. Since this file is a .csv (comma separated value) file a comma was chosen.

Text From List

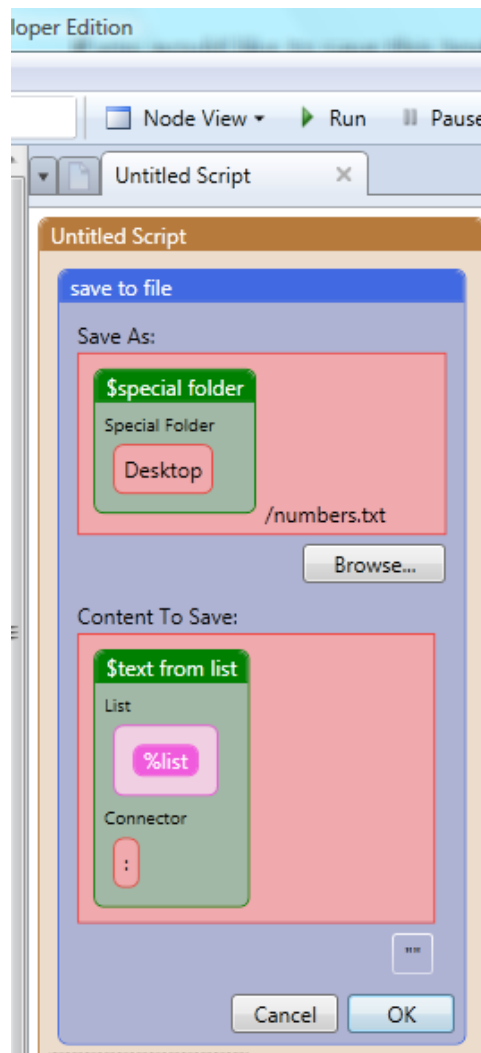
This function creates a text string based on a list. It will connect the list items with the specified connector. It works in exactly the opposite manner as \$list from text.

The list that was created from the .csv file in the previous example can now be saved to a new file using any delimiter you choose.

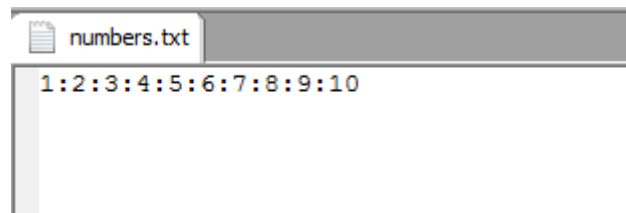
For example, the original .csv file, if opened in a text editor looks like this:

1,2,3,4,5,6,7,8,9,10

If you would like to save this text in a new file using “:” as a delimiter, you would simply implement the \$text from list function to achieve this.



The first step is to bring the “save to file” command into the scripting area. Provide the location and name of the file you wish to save to. From there use the \$text from list as the function and choose the list you wish to use. Provide a delimiter (in this case the “:”). That’s it. The new file looks like this:



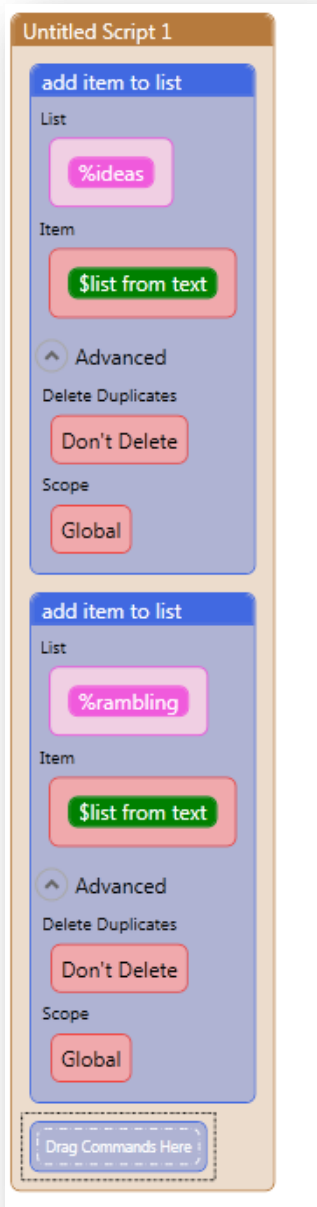
Subtract lists (PRO Edition Only)

This command allows you to compare two lists and create a new list containing all items that are unique to the first list.

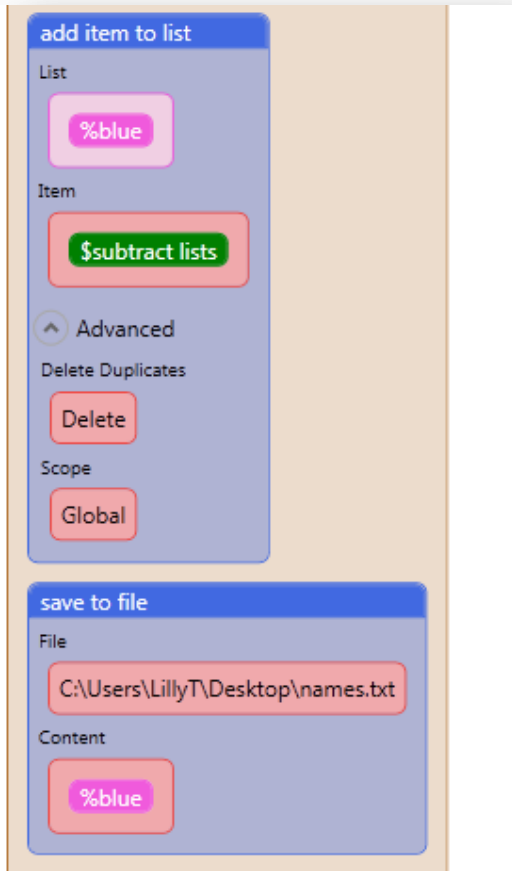
In this example, we have two lists. List one, which is called \$ideas, contains the following: a,b,c,d

The other list, called \$ramblings contains the following: a, b, g,h

Notice that the items in the list are added to the list with the list from text function, which allows you to type in your list items as opposed to adding a list into UBot from a file on your computer with the list from file function. To make the lists, we are using the add item to list command.



What we want to do is to subtract the list \$ideas from the list \$ramblings, and leave only the items that are unique to only the first list. We then want to save the unique items to a file on our computer.



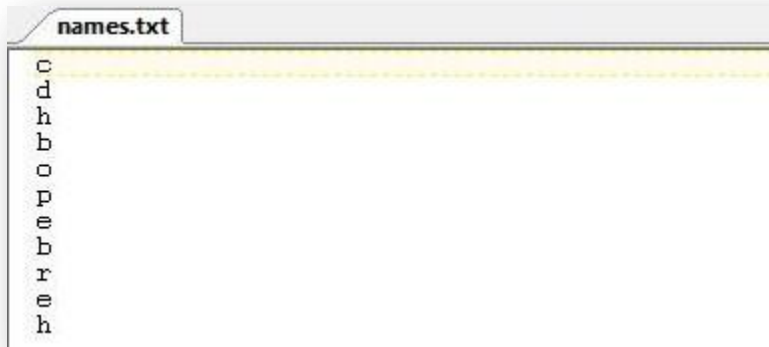
When the script is run, the following results are saved to the file:



Sort lists (PRO Edition Only)

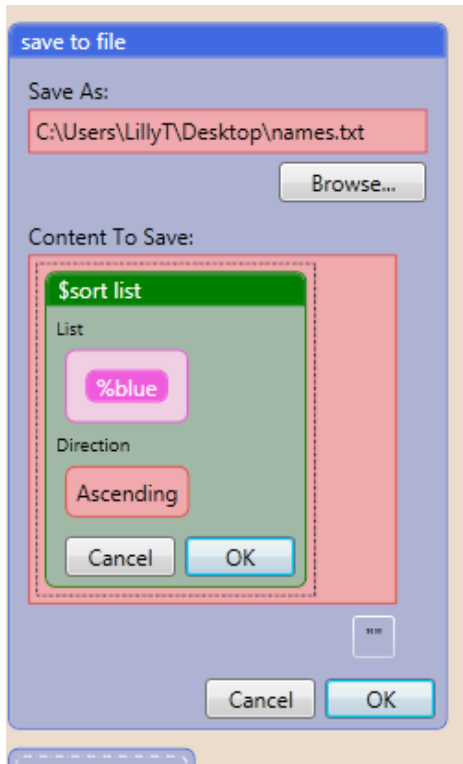
Sort List allows you to sort all items contained in a list in either ascending or descending order.

In this example, I have a list of letters:

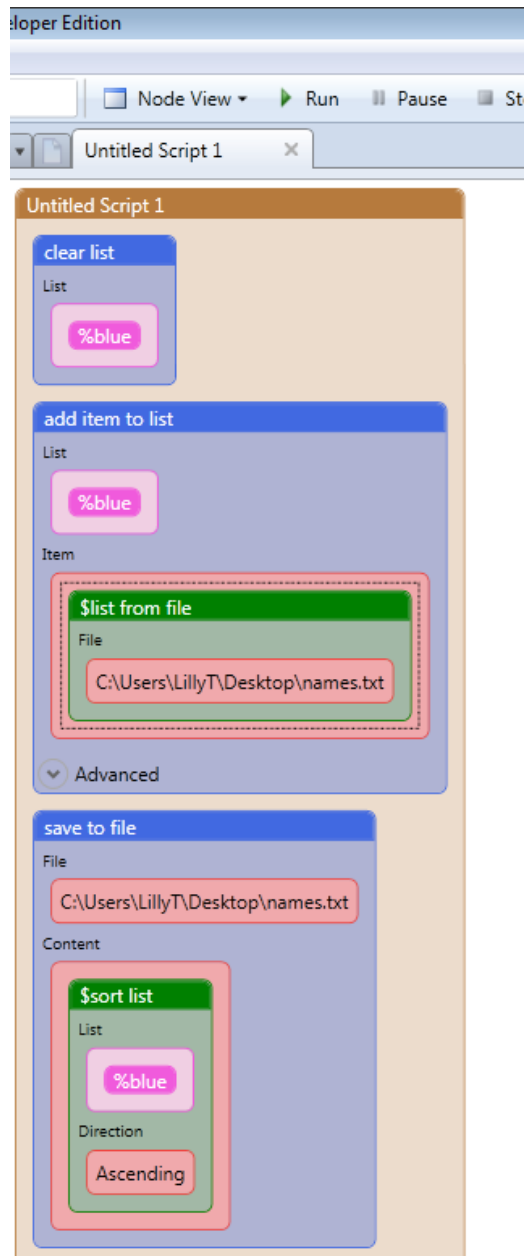


```
names.txt
c
d
h
b
o
p
e
b
r
e
h
```

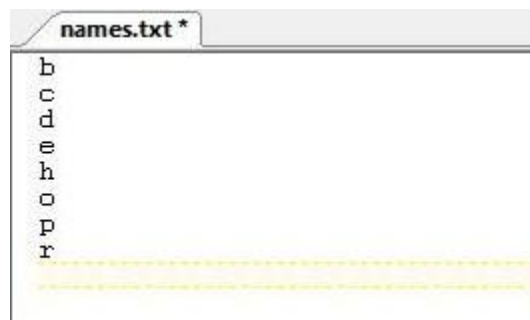
We are going to sort this list in ascending order. The list is added from a file with the list from file function. We are sorting the file while we are saving the items to a file. In the save to file command, instead of inserting the list directly into the save to file command, you would insert a sort file function, and then choose the list containing the items you would like to sort.



This is the sort list function will look like this after being dragged into the save to file command. A list is chosen for sorting within the function.



When the list is sorted, the result will be saved to the file.



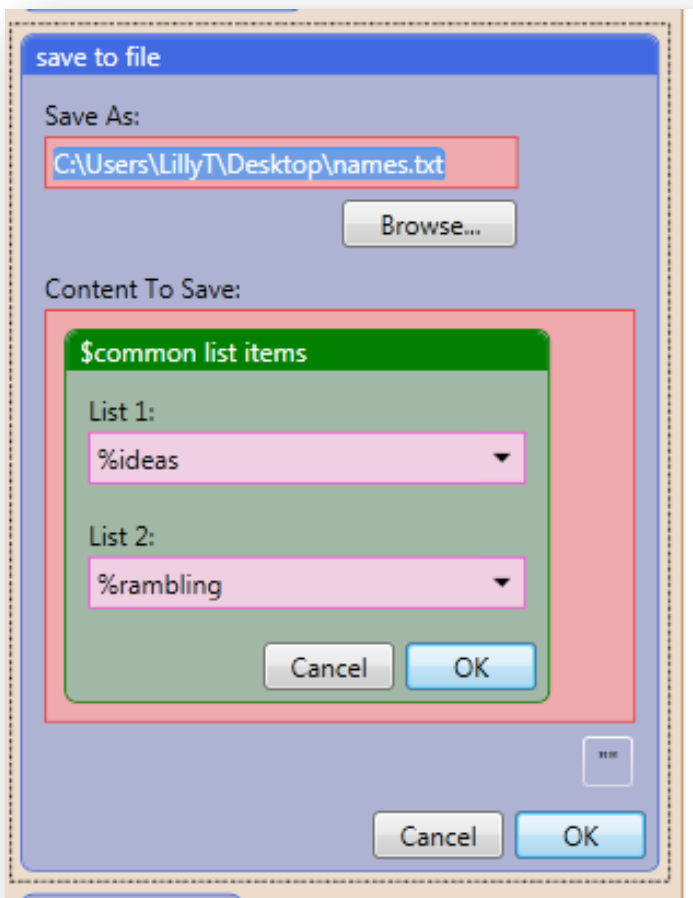
Keep in mind that if you set Delete Duplicates to Yes in your add to list command, duplicate items will be removed from the list.

Common list items (PRO Edition Only)

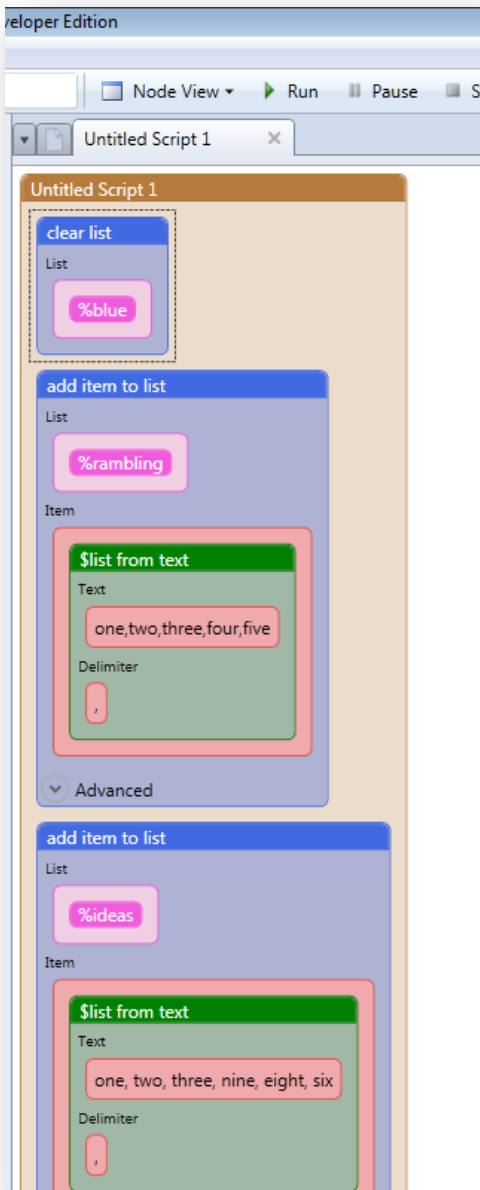
This function allows you to compare two lists and create a third list containing all items that are common to both lists.

This command works in the opposite way that the subtract list function works. Instead of comparing two lists and finding items only unique to the first list, it finds list items common to both lists.

In this example, we have two lists, list one and list two.



We are then finding the common list items in the save to file command, and then saving the common items in a file.

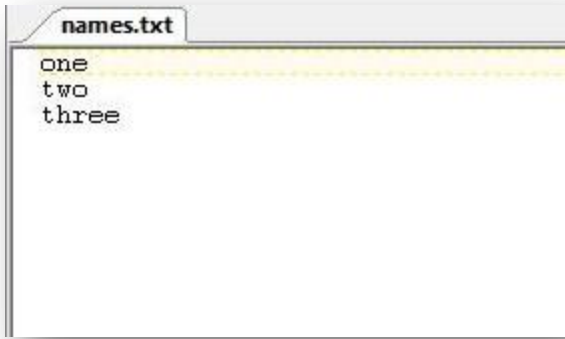


The image shows a workflow editor with two main steps:

- add item to list**:
 - List**: %ideas
 - Item**:
 - \$list from text**:
 - Text**: one, two, three, nine, eight, six
 - Delimiter**: ,
 - Advanced**: (collapsed)
- save to file**:
 - File**: C:\Users\LillyT\Desktop\names.txt
 - Content**:
 - \$common list items**:
 - First List**: %ideas
 - Second List**: %rambling

At the bottom, there is a button labeled "Drag Commands Here".

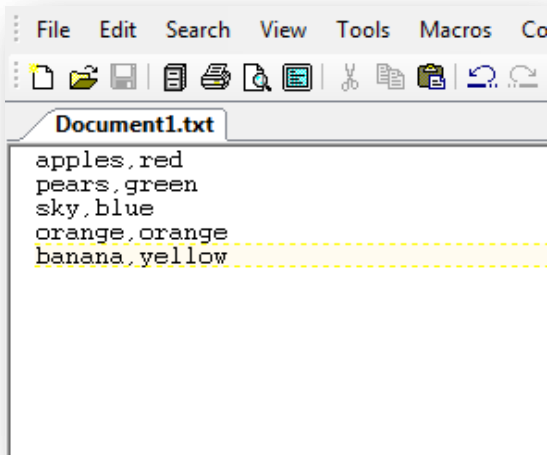
And this is the resulting items when the common items in the list are done.



Because the words one, two and three are common to both lists.

Table Cell

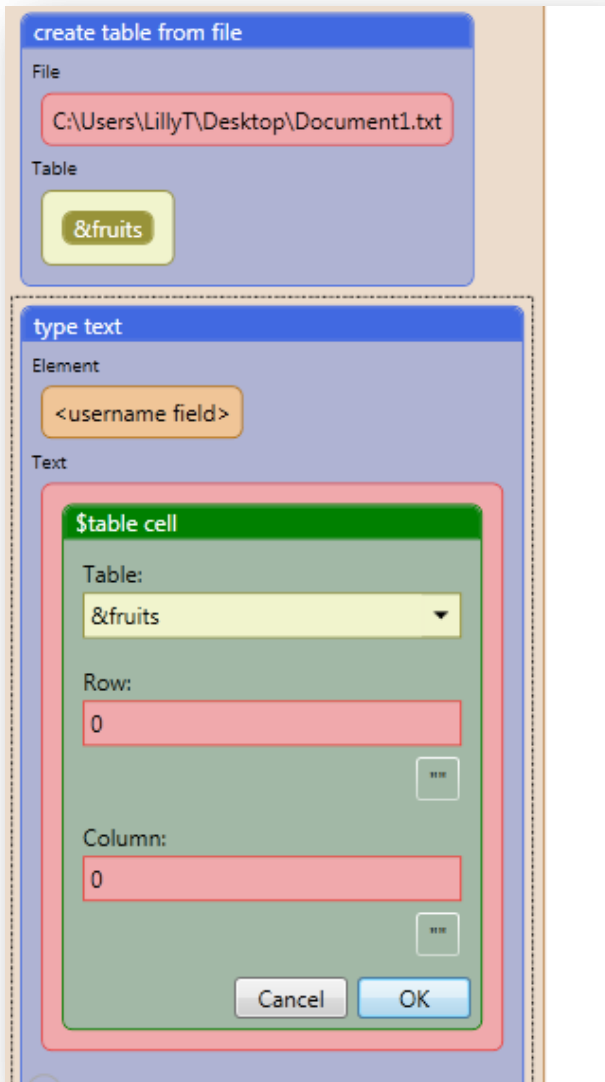
This command will call the contents of a specified table cell. In this example, we have a csv file with information; fruits in one column, and their corresponding colors in the other column. The information is comma separated.



There are 2 columns and five rows.

We are going to create our table by dragging the create table from file command from the Data command, browse for our csv file and then name our table.

Afterwards, we will drag the field we want to fill into the scripting area. In the content area of the type text command that emerges from dragging the field into the script, we will insert the table cell function. For this example, we are filling the username field with the contents in column 0, row 0.



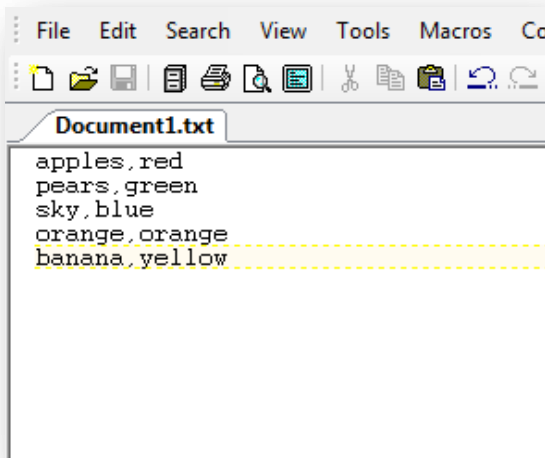
When we click ok on the commands and run the script, our field is filled with the contents of column 0 row 0, which was the word "apple".

username:

Table Total Rows

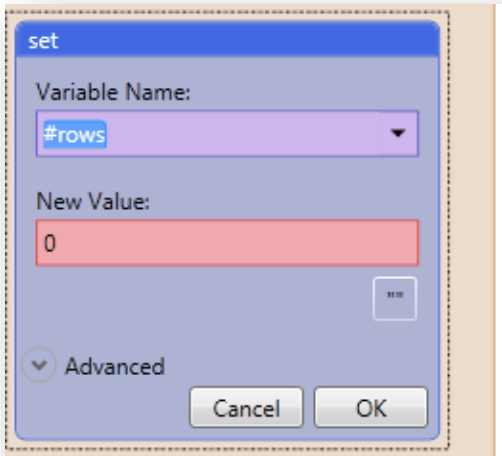
This command will grab the total rows within a table. This is especially useful within a loop, when you are trying to fill a lot of fields with the contents of a certain column of information.

In this example, we have created our table from the following file.



We are trying to fill a field with the contents of the left column. In this case, we will need to create a loop that loops according to how many rows we have within the table. In this case, there are 5 rows, and so the loops will run 5 times. Inside the loop is a type text command from the field you want to fill, with a table cell function.

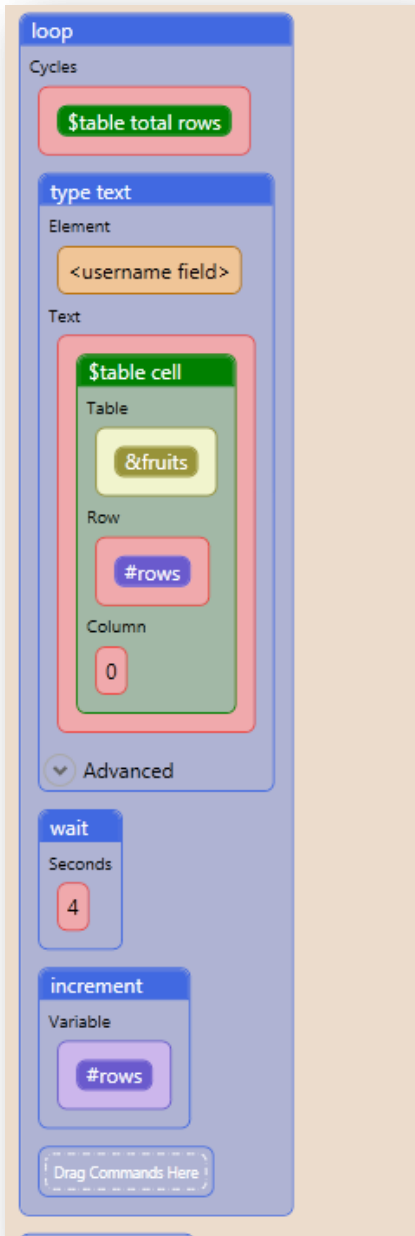
Before we move on with the table cell function, we will create a set command outside of the loop, called the variable "rows" and set the variable to 0.



We will be placing that variable “rows” into the table cell command, under rows.

Click ok to finish editing.

Afterwards, we will add a wait command for 3 seconds. Finally, we will add an increment command from the data commands, and increment the variable “row”.



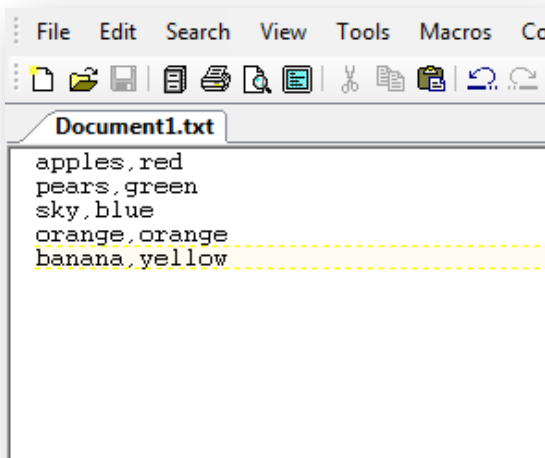
When we run our script, the field you chose will be filled with each item from the left column of the table.

Table Total Columns

This command will grab the total columns within a table. This is especially useful

within a loop, when you are trying to fill a lot of fields with the contents of a certain column of information.

In this example, in the file we created in our last example for table total rows, we want to fill a field with the contents of the first row in the file.



We would use the same setup we had in the example for table total rows, except the variable will go under the area labeled "column" instead of the area labeled "row" in the table cell function. Also, the loop will loop according to the table total columns.

Your script should look like this:

loop

Cycles

\$table total columns

type text

Element

<username field>

Text

\$table cell

Table

&fruits

Row

0

Column

#rows

Advanced

wait

Seconds

4

increment

Variable

#rows

Drag Commands Here

Feel free to rename the variable to avoid confusion. The name of the variable being incremented will not affect the process within the script. It is just a matter of how you would like to keep track of everything within your script.

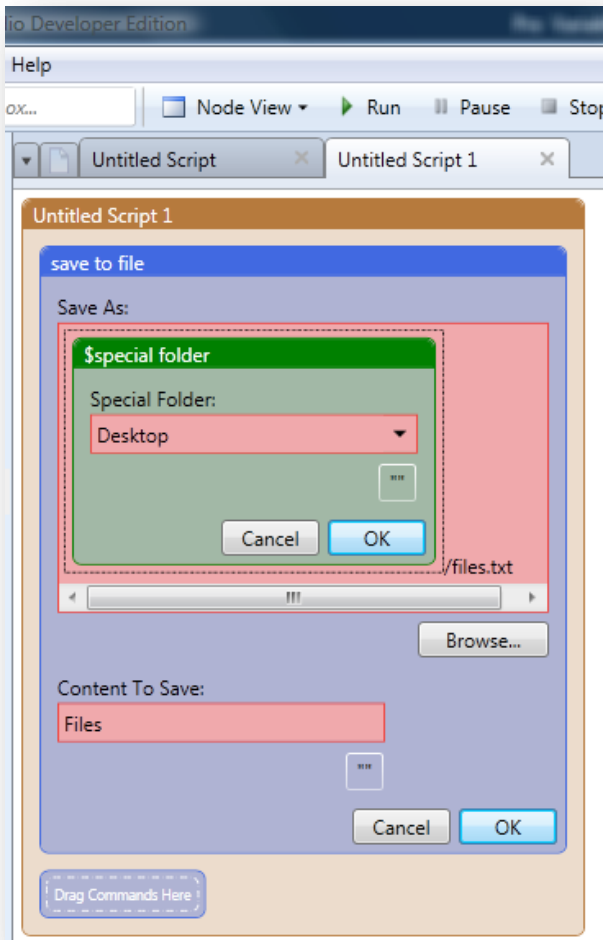
When the script is run, the field you chose to fill should be filled with the contents of the first row of the table. To change the row, simply change the number under the “row” field in the table cell function.

File Functions





Special Folder

This command will allow you to save or grab a file from or to the Application Data folder, the Desktop, Program Files, Documents, and many more on the computer the bot is run on. Simply select your option from the drop down menu in the command.

In this example, we are using the save to file command to save a file onto some location on a person’s computer.



We have chosen to save a file called files.txt to the desktop, no matter what computer we're on. If we run the script, the file "files.txt" will be found on the desktop of the computer we ran the script on.

Name	Size	Type	Date modified
	2.04 KB		 1.90 KB
 Windows Media Player Shortcut 916 bytes			 files TXT File 5 bytes

Get files (PRO Edition Only)

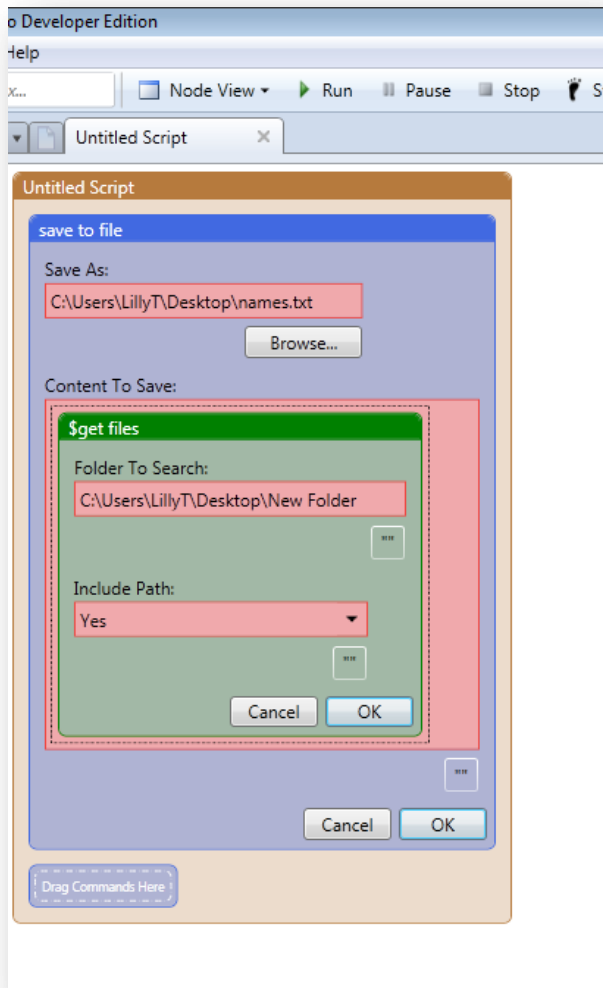
Files From Folder returns a list of all files in any given directory.

In this example, we will be using a save to file command. We will be saving the file names, with and without the file paths of the contents of a folder.

First, drag in your save to file command and browse for the file you'd like to save the contents of the folder to.

In the area in the save to file command designated "Content", go under the File functions under Functions and drag in the get file function. Two areas will open up within the function. The area designated Folder is where your folder path will go.

The area Labeled "Include file path" is where you will determine whether or not you want the file path included in the file names. For the first example, we will select "Yes".



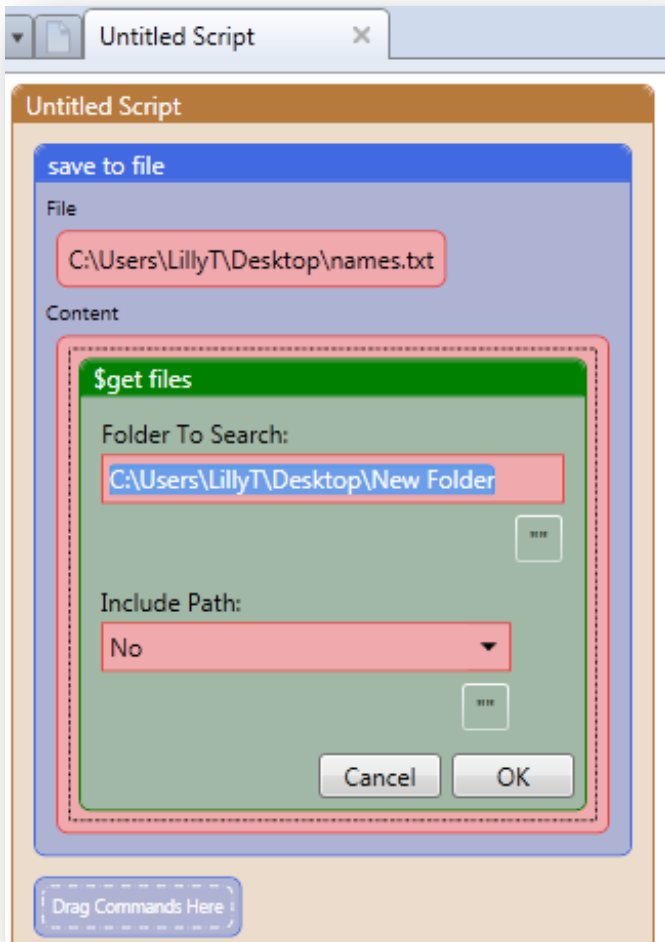
Click ok for the get files function as well as the save to file command.

When you run the script, the file you selected in the save to file command will be filled with the names of the files in the folder you designated in your get files function.

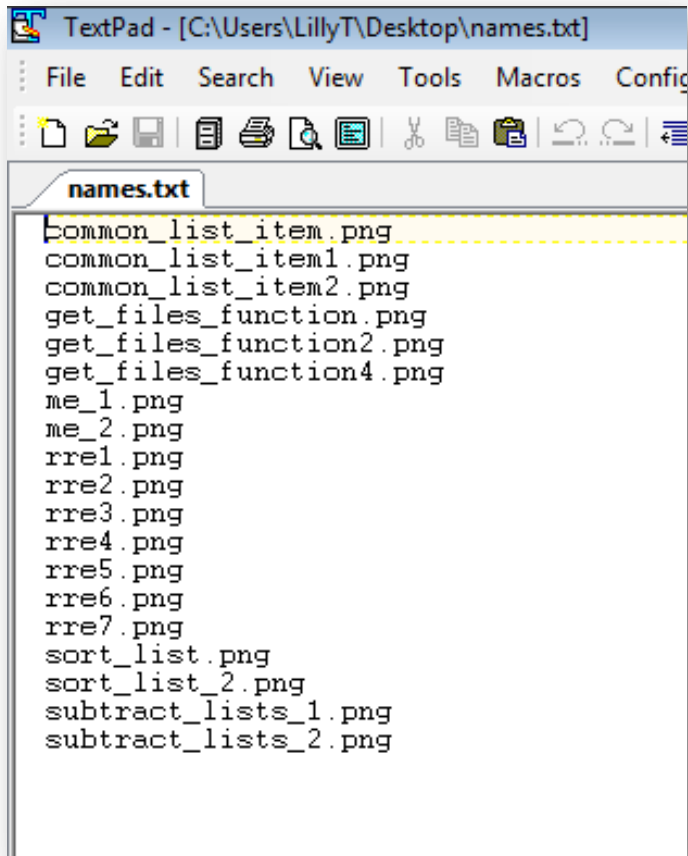
```
names.txt
C:\Users\LillyT\Desktop\New Folder\common_list_item.png
C:\Users\LillyT\Desktop\New Folder\common_list_item1.png
C:\Users\LillyT\Desktop\New Folder\common_list_item2.png
C:\Users\LillyT\Desktop\New Folder\get_files_function.png
C:\Users\LillyT\Desktop\New Folder\me_1.png
C:\Users\LillyT\Desktop\New Folder\me_2.png
C:\Users\LillyT\Desktop\New Folder\rre1.png
C:\Users\LillyT\Desktop\New Folder\rre2.png
C:\Users\LillyT\Desktop\New Folder\rre3.png
C:\Users\LillyT\Desktop\New Folder\rre4.png
C:\Users\LillyT\Desktop\New Folder\rre5.png
C:\Users\LillyT\Desktop\New Folder\rre6.png
C:\Users\LillyT\Desktop\New Folder\rre7.png
C:\Users\LillyT\Desktop\New Folder\sort_list.png
C:\Users\LillyT\Desktop\New Folder\sort_list_2.png
C:\Users\LillyT\Desktop\New Folder\subtract_lists_1.png
C:\Users\LillyT\Desktop\New Folder\subtract_lists_2.png
```

So what if you want the file names without the file path?

Simply right click the function and select edit from the right click menu. Go to the area where you are asked include or exclude the file path. Click the small drop down arrow and select "No". Click ok on the function, and then run your script.



The file you selected in the save to file command will be filled with the names of the files in the folder you designated in your get files function.



The image shows a screenshot of a TextPad window. The title bar reads "TextPad - [C:\Users\LillyT\Desktop\names.txt]". The menu bar includes "File", "Edit", "Search", "View", "Tools", "Macros", and "Config". The toolbar contains icons for file operations like opening, saving, printing, and searching, as well as editing functions like cut, copy, paste, and undo. The main text area, titled "names.txt", contains a list of 18 PNG file names:

```
common_list_item.png
common_list_item1.png
common_list_item2.png
get_files_function.png
get_files_function2.png
get_files_function4.png
me_1.png
me_2.png
rre1.png
rre2.png
rre3.png
rre4.png
rre5.png
rre6.png
rre7.png
sort_list.png
sort_list_2.png
subtract_lists_1.png
subtract_lists_2.png
```