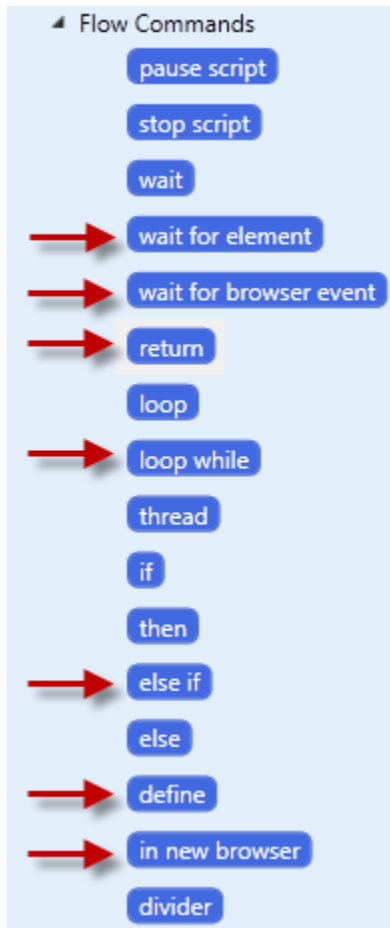


The UBot Studio

SCRIPT REFERENCE

The Flow Commands

In this script reference, we will cover the Flow commands in this section which are located in the third sub-menu of the toolbox on the left side of the dev tool:



In the image above you can see all 16 flow commands available in Version 4.

Pause Script

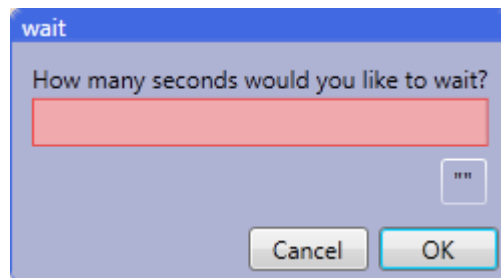
This command will pause the script and is generally used for debugging purposes. If you choose to add this command to your script, simply drag it anywhere in the scripting area where you would like the script to pause. Once paused you can resume the script by pressing the play button again.

Stop Script

This command Stops the script from running. Like pause, you can place it anywhere in your script where you would like to stop the execution of commands.

Wait

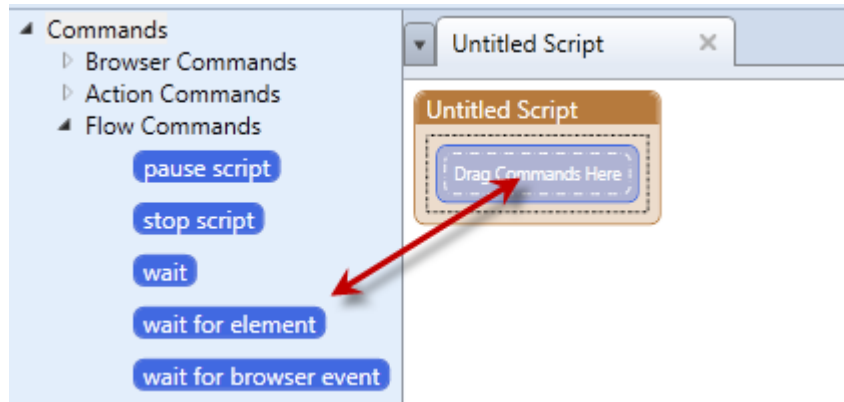
This command replaces the old “delay” command in v3.5. When dragged into the scripting area, you will see the parameters:



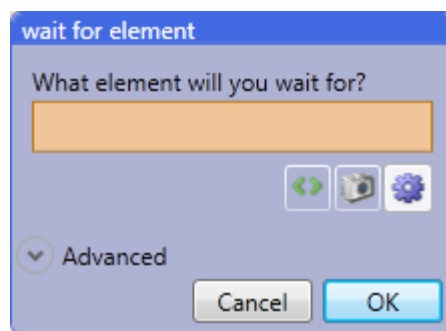
This command is self-explanatory and only requires you to enter the number of seconds you wish to wait prior to executing the next command. Enter the number in the pink box and click ok.


Wait for element

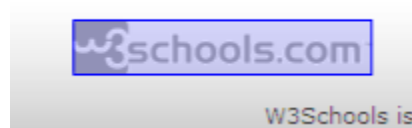
This command is also relatively self-explanatory as it allows you to delay the script until an element on the page is present. This command is based on the selector tools which you can choose from to select an element on the page (See the section titled “The Version 4 selectors” for more information on how to use them). You will first drag this command into the scripting area:



At which time you will be presented with the parameters box:



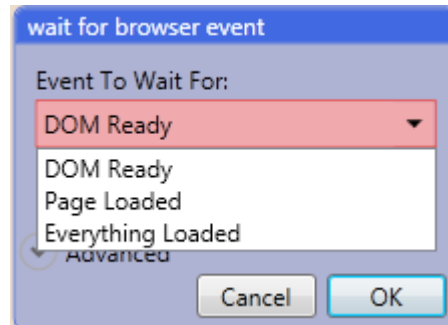
You are now ready to choose a selector and then choose an element in the browser page. For this example I will choose the element selector () . In this case I will scroll to the bottom of the page and choose the site's logo:



Once chosen, click OK and you are done with this command. The next command will not execute until your chosen element has fully loaded on the page.

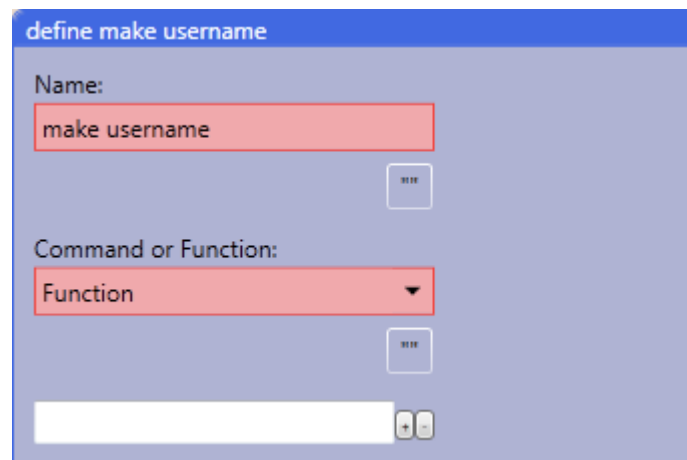
Wait For Browser Event

This command waits for a browser event to happen such as a page being loaded and DOM ready. When you drag the command into the scripting area, simply choose one of the three options and click OK:

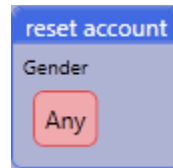


Return

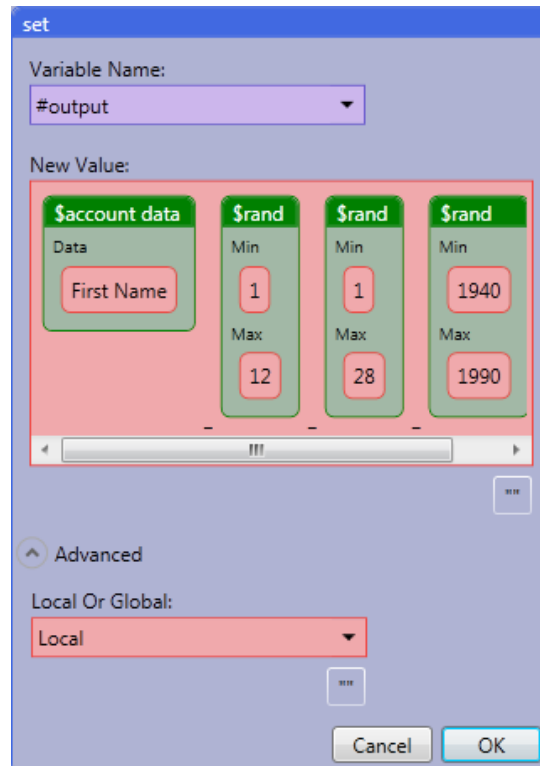
This command exits the current command or function and allows custom functions to return a value. This command is designed to allow the user to re-use code in the form of custom functions where a value is returned. In this example we will create a custom function that generates a custom username including a first name and a date of birth. First, drag the command “define” into the scripting area, give it a name and choose function from the dropdown:



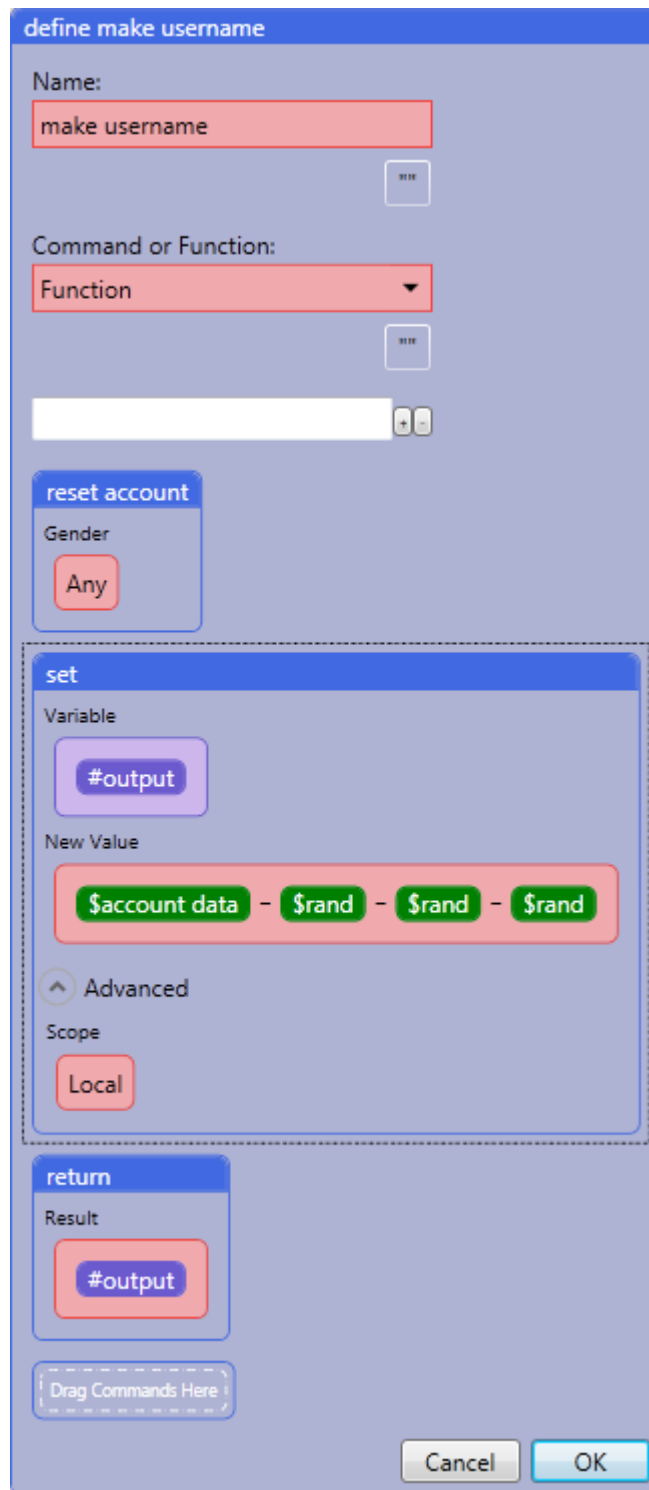
You can now drag a “reset account” command into the define node and leave it as “any”:



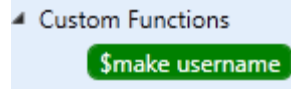
From here drag a “set” command into the node. In this case we are naming the variable #output. The value includes a first name (account data) and three \$rand nodes to create the random date of birth. They are all separated by underscores (_) not dashes:



Clikc OK and drag a \$return command into the node:

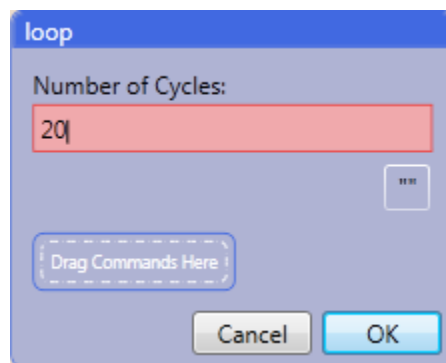


Click OK and you are done. Anytime you need a custom username simply call the custom function you created from the custom function menu:

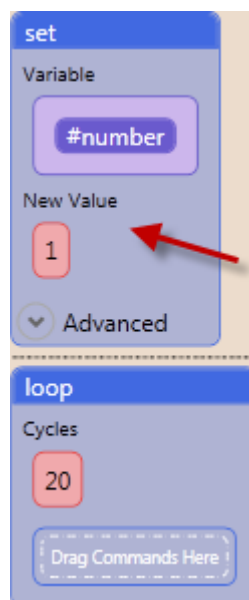


Loop

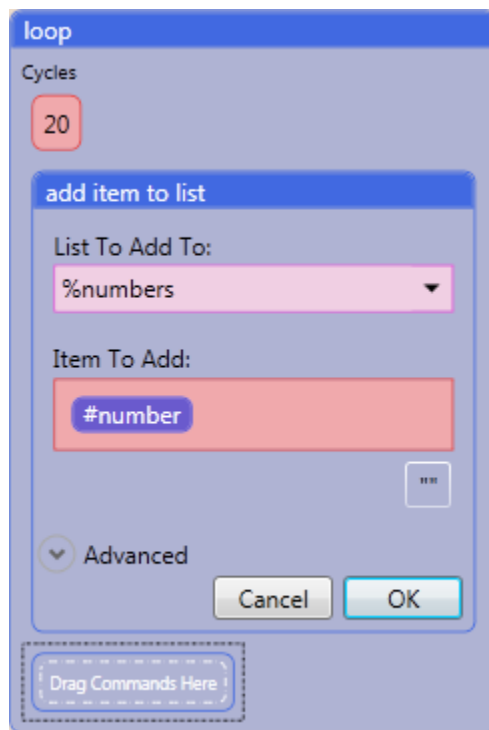
This command runs the contained commands a specified number of times. It allows you to repeat tasks without coding them multiple times. This example will show you how to add 20 items to a list using a loop. First drag the loop command into the scripting area and input the number of cycles you want it to complete. In this case we will enter 20:



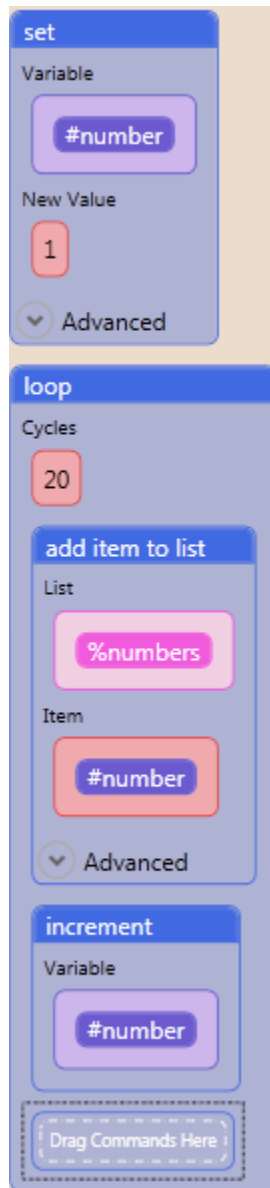
Click OK and you can begin adding commands inside the loop. For this example, we will be setting a variable outside the loop. We will call it #number and assign it a value of 1:



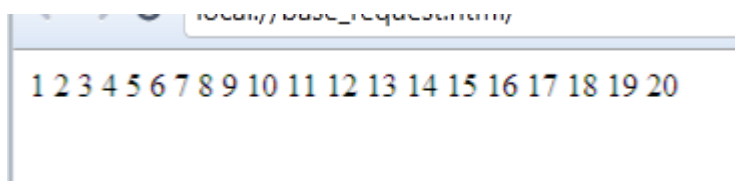
Now, inside the loop we will need to drag the “Add Item To List” command. Call it numbers, and add the variable you just created as the content:



Click OK and then add an increment command to increment the variable you set:

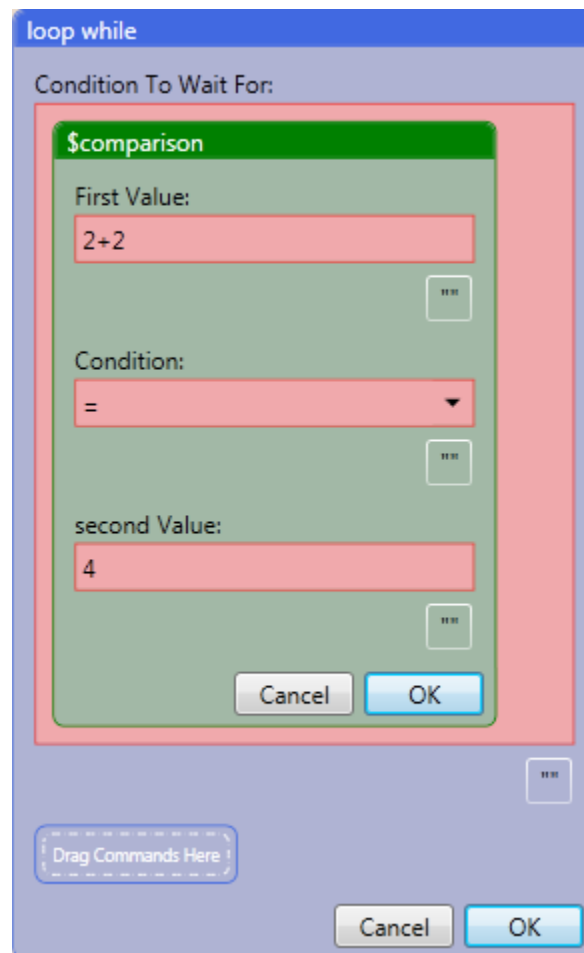


When you run this script, you will create a list with 20 items (numbers 1–20):



Loop While

This command runs the contained commands as long as a condition returns true. You can create any condition you like, for example, you could specify that as long as a certain element \$exists on a page execute the commands in the loop. Or using the above example, you could say, as long as #number is less than 15 (<15), execute the commands in the loop. There are numerous ways you can set up a while loop as the condition you set can be almost anything. We will take a look at a simple example. First drag the while loop into the scripting area and add a \$comparison node into the condition to look for box:



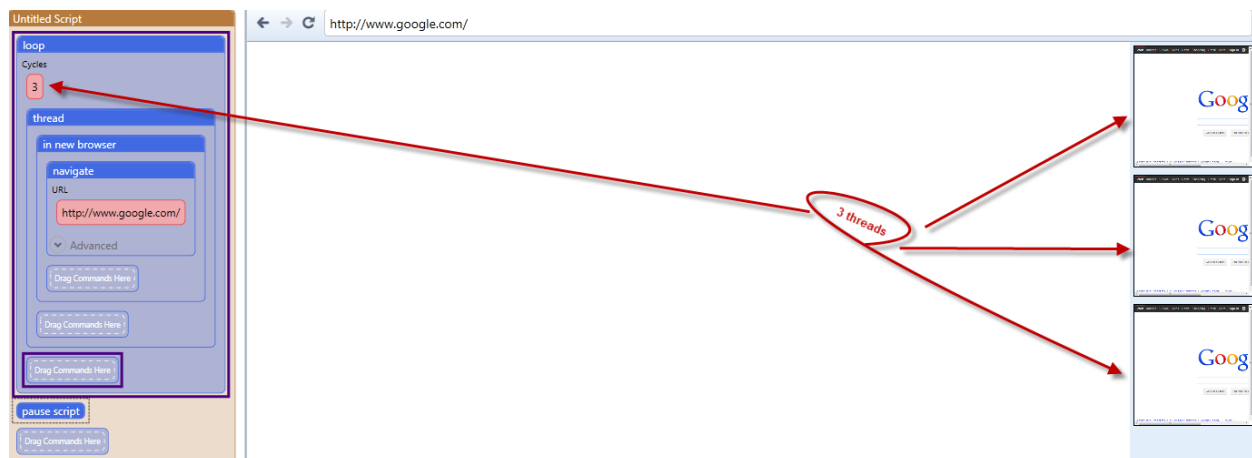
This condition states, "As long as $2+2=4$, run the contained commands". Obviously that loop would be infinite as $2+2$ will always equal 4, but the conditions you set will not be.

Thread

This command runs the contained commands in a separate thread from the main script allowing multiple commands to run at the same time. In this example we will set up a small multithreading script. The initial setup will look like this:

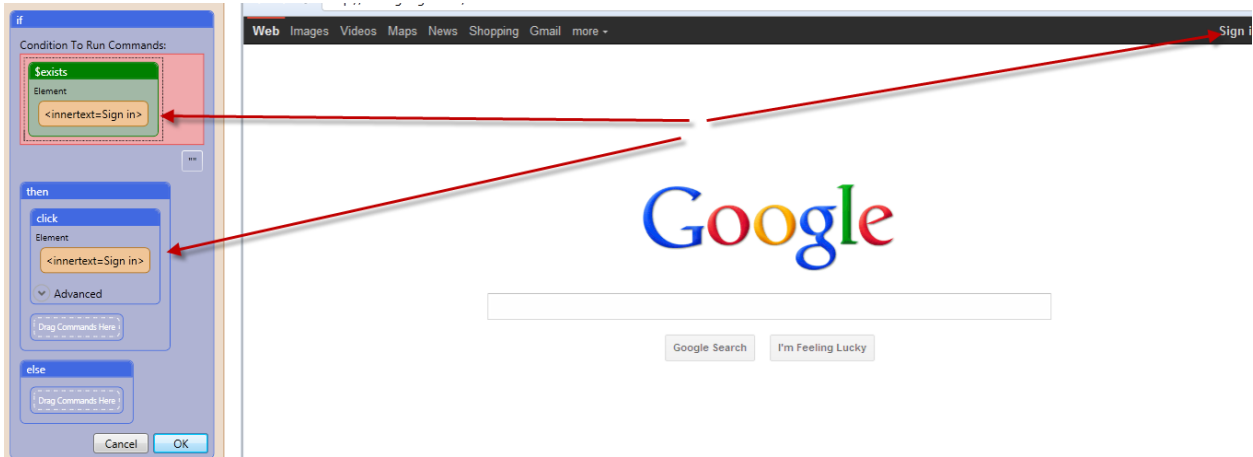


Every command placed inside the “in new browser” command will be executed in an independent thread. This particular example is using 3 threads that will execute simultaneously. We will add a simple navigate command inside the node and it will execute three times simultaneously when run (It will have three separate instances of ubot navigating to that site):

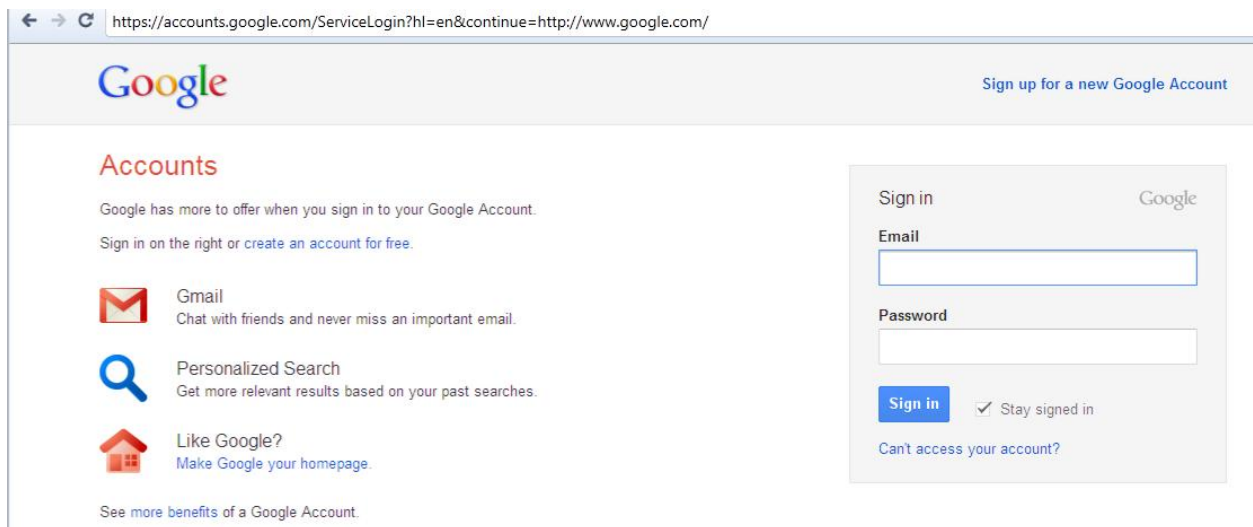


If

This command runs the contained commands if the condition is true. The following example simply states that IF the sign in button is present, THEN click the sign in button:



When the script is run it will click the sign in button and navigate to the login page:

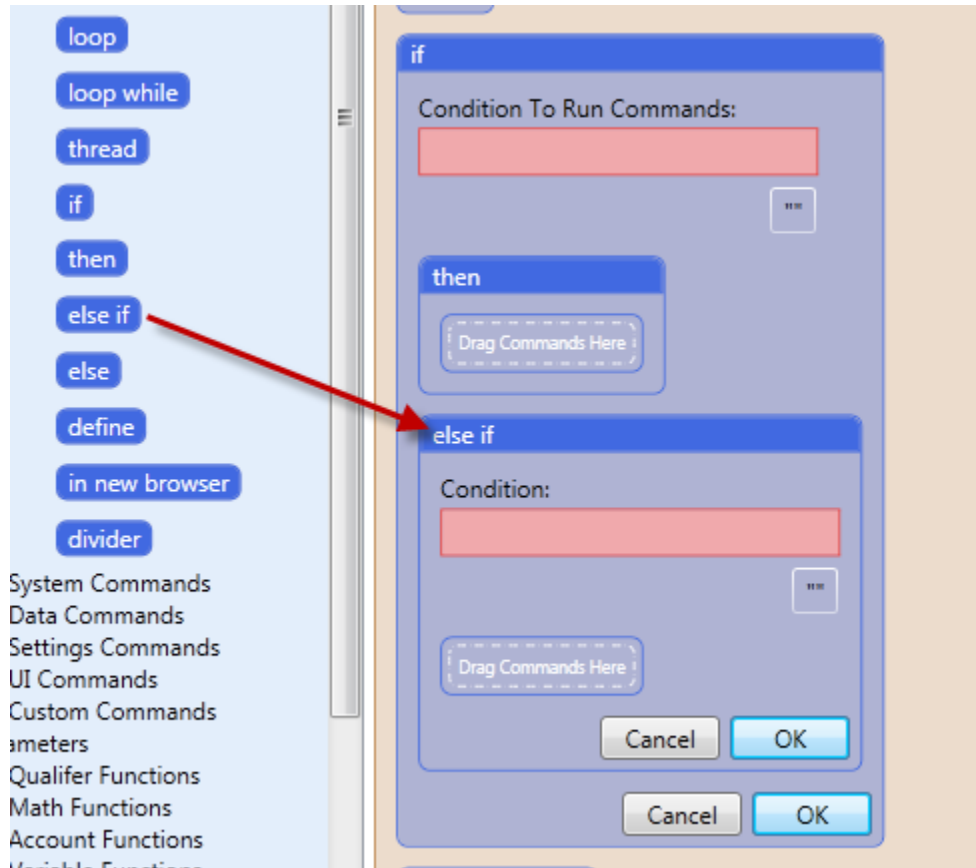


Then

This command is used inside an if command to specify which commands should run if the condition is true. The If command already has an Then command included and you should not have a need for this command very often. It is, however, available should the need arise.

Else If

This command is to be used inside an If statement to check for another condition if the first condition was not returned true. This is an enhancement in version 4 that helps avoid nesting If statements inside each other. You can simply replace the Else node in an If statement with the Else If command:

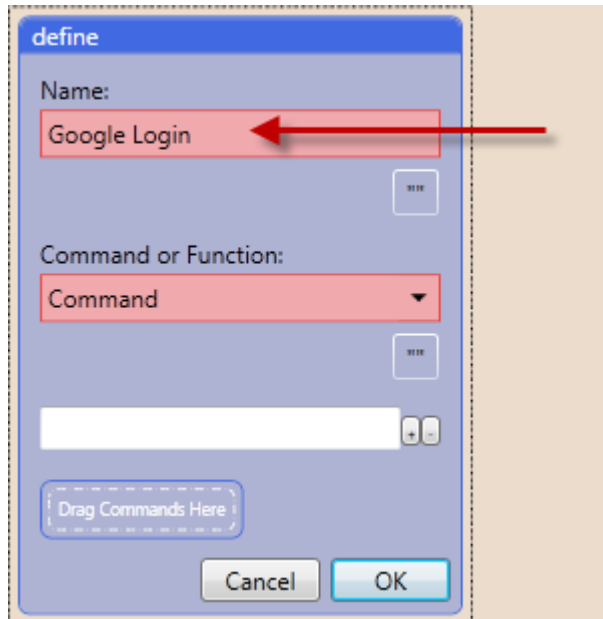


Else

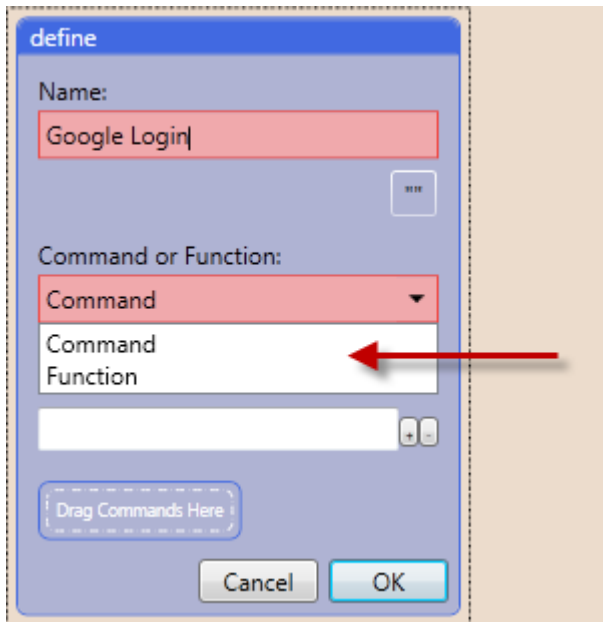
This command is used inside an if command to specify which commands should run if the condition is not true. The If command already has an Else command included and you should not have a need for this command very often. It is, however, available should the need arise.

Define

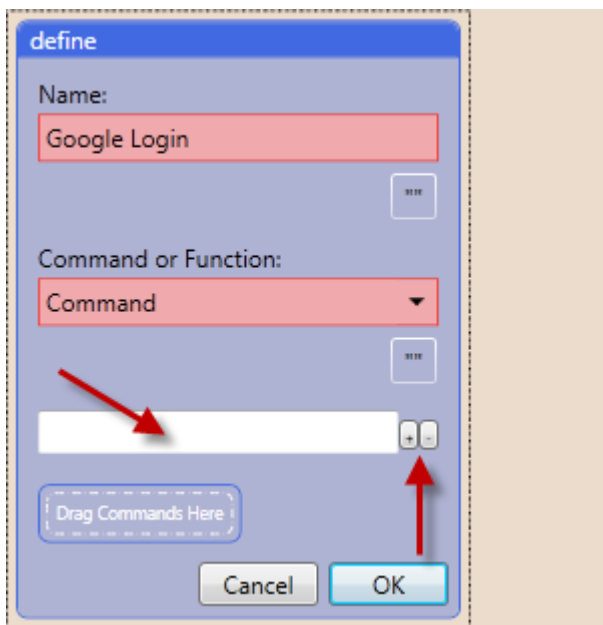
This command allows you to define a custom command or function that can be used in your bot. This command replaces the sub command in v3.5, but it works in a very similar way. When you drag this command into the scripting area you will be required to provide a name (i.e. "Google Login"):



The second option requires you to choose whether this will be a command or a function:



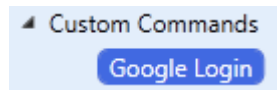
The third option allows you to add parameters if you need to do so. There are two buttons marked with a “+” and “-” that allow you to add or remove parameter fields:



Click OK and you can now start adding commands to your custom command/function. In this example the custom command will just navigate to Google and log in:



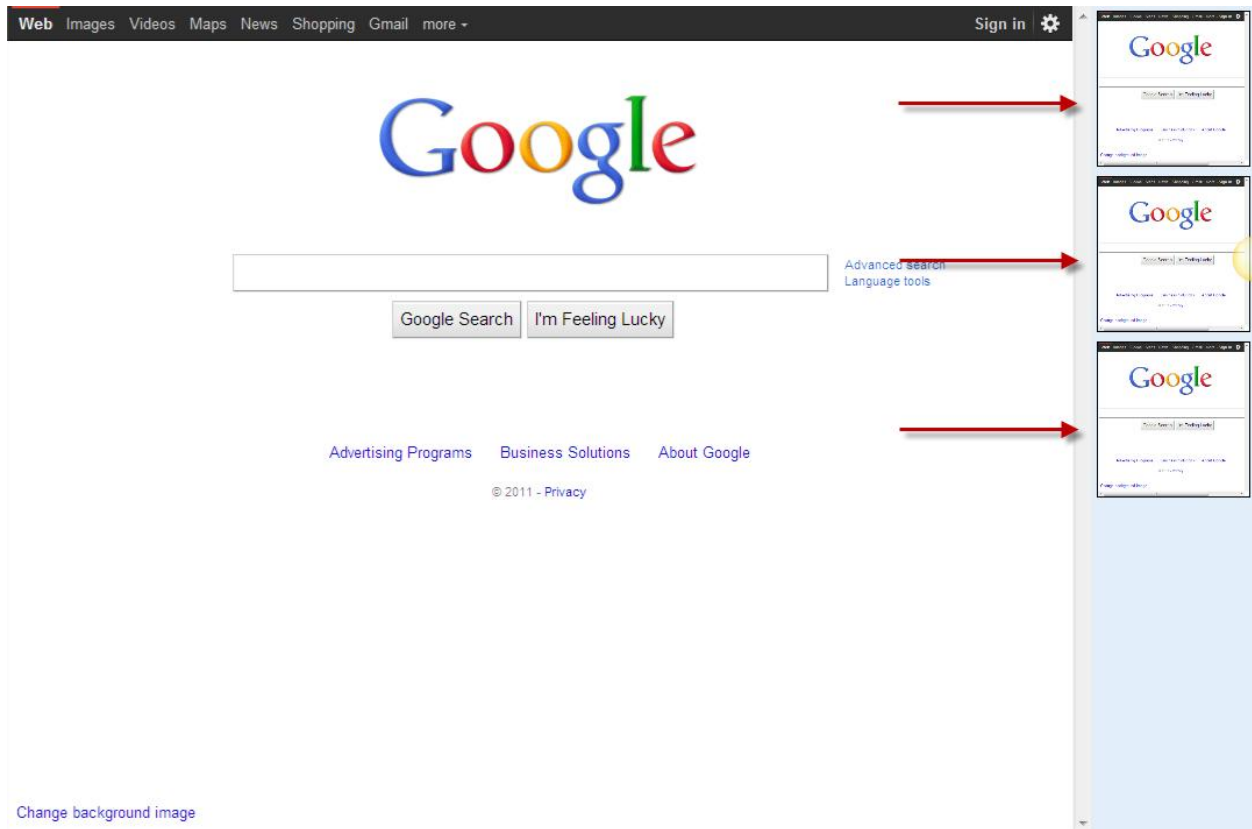
Now that you've completed your custom command, you will find it under custom commands:



Just drag that custom command into the scripting area where you want the command to run. You may notice that it is similar to the “run sub” command from v3.5. That is because it works in much the same way. So you can consider your custom commands/functions serve the same purpose as a “run sub” command.

In New Browser

This command runs all contained commands inside a separate browser in a separate window. In version 4 of UBot Studio, these separate browsers will appear as small browsers to the right of the main browser in a vertical column:



Simply drag the “in new browser” command into the scripting area, and place all desired commands inside that node:



Divider

This command visually divides up the code in the scripting area to assist in the visual organization of code. By dragging this node into the scripting area, you will be placing a divider line between pieces of code:

