Table 1

	1	2	3	4	
Simply call a method of the host object	'method_name'				#1 arg → check if the method_name exists in host or globals
	<function>				
Same, but put everything in a tuple	'method_name'				#1 arg → check if the method_name exists in host or globals
	<function>				
Call a method of a class	'method_name'	ClassHolder			#2 args → Check if 2nd argument is a class name, then first argument is method_name, and second argument is the class_name.
	<function>	<class>			
Call a method of the host object, and keep the result in a new attribute	'new_attribute'	'method_name'			#2 args → If both elements in tuple are strings, 1st argument is attribute_name and 2nd argument is method_name
	<string>	<function>			
Call a method of the host object, with specific parameters	'method_name'	{'param1': 'value1'}			#2 args → Check if 2nd argument is a dictionary, then first argument is method_name, and second argument are parameters.
	<function>	<dict>			
Call a method of the host object, with specific parameters, and keep the result in a new attribute	'new_attribute'	'method_name'	{'param1': 'value1'}		#3 args → If 1st and 2nd argument are both strings, then we have a tuple with attribute_name, method_name and parameters.
	<string>	<function>	<dict>		
Call a method of a specific class, with specific parameters.	'method_name'	ClassHolder	{'param1': 'value1'}		#3 args → If 2nd argument is a class, 1st argument is method_name, 2nd argument must be a class_name, and 3rd argument are the parameters
	<function>	<class>	<dict>		
Call a method of a specific class, with specific parameters, and keep the result in a new attribute	'new_attribute'	method_name'	ClassHolder	{'param1': 'value1'}	#4 args → We've attribute_name, method_name, class_name, parameters
	<string>	<function>	<class>	<dict>	

1