Team 6: Jython Automated Testing Framework

MH Johnson , Henry Noonan

What Is Jython

Jython is an open source programming language. It is an implementation of the Python language which runs on the Java Virtual machine, as such it falls under the umbrella of the Python Software Foundation.

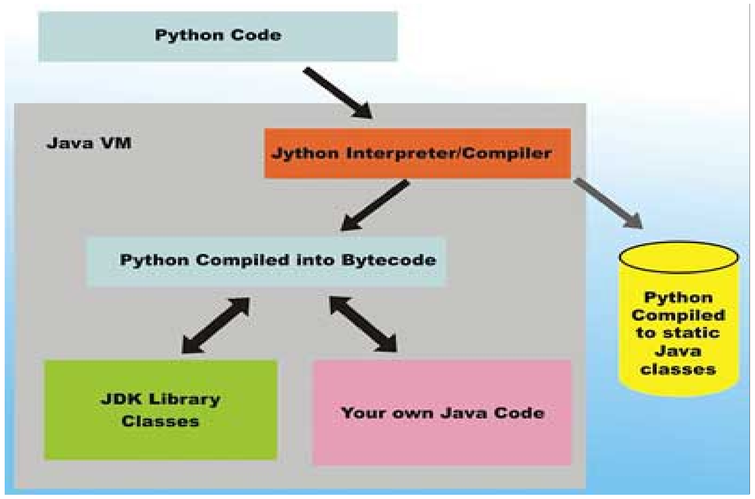

Figure 1.Simple diagram of the Jython Architecture
Taken from http://daveti.blog.com/2012/11/28/jython-combination-of-java-and-python/

Testing Plan

Our Testing Framework was planned out with the goal of running a suite of 25 test cases on the Jython codebase. The framework was designed so as to be as modular as possible, parsing each test case from its own text file, such that test-case could be quickly written, and specify a different library, function and arguments, as well as the expected output, which for our framework must be specified manually

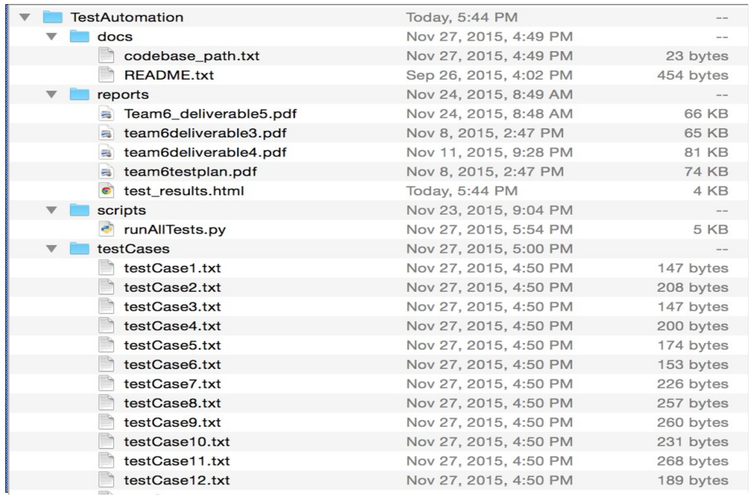

Figure 2. The Framework's file organization

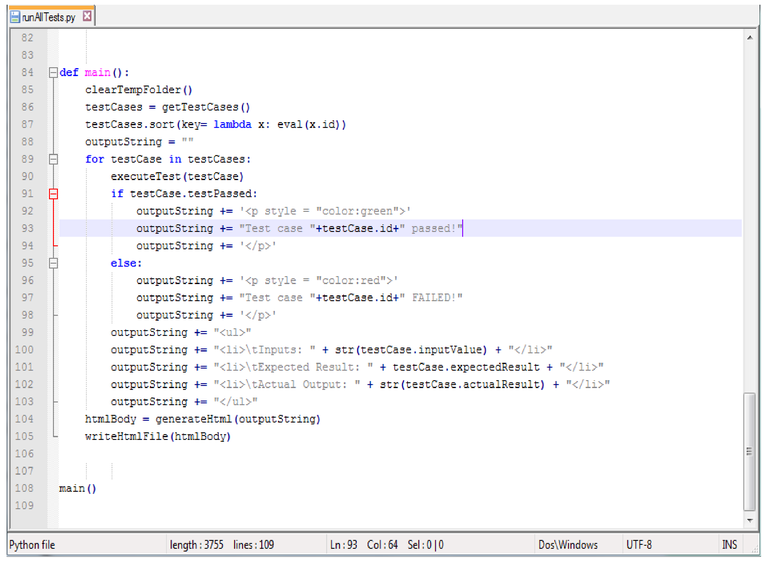

Figure 3. Main method of the Automated Testing Framework Script

Testing Framework

The script, whose main method is pictured above (figure 3). Clears the necessary folders for its operation, and subsequently iterates through the files in the test case folder, which are parsed and run as tests according to the test case template we have developed (figure 5). After parsing a test, the script makes the specified calls to functions in the codebase along with the specified arguments, and compares these results to the expected outputs (also parsed in) as seen in figure 4.

Test Results

Test case number	Status	Method Being Tested	Inputs	Expected Result	Actual Result
1	Passed!	upper(s)	['caT']	CAT	CAT
2	Passed!	strip(s)	[' this is a test for trim ']	this is a test for trim	this is a test for trim
3	Failed!	lower(s)	['CAT']	cat	CAT
4	Passed!	swapcase(s)	['tESTING FOr Cases']	TEsting FoR cASES	TEsting FoR cASES
5	Failed!	capwords(s)	['this iS a tEst.']	This Is A Test.	this is a test.
6	Failed!	unquote(s)	['abc%20def']	abc def	abc%20def
7	Passed!	replace(s, old, new)	['here is a test', 'test', 'TEST']	here is a TEST	here is a TEST
8	Passed!	replace(s, old, new)	['she sells seashells by the seashore', 'she', '']	sells sealls by the seashore	sells sealls by the seashore
9	Passed!	find(s, *args)	['looking for the first instance of the letter g', 'g']	6	6
10	Passed!	count(s, *args)	['example string with a good number of characters in it', ' ']	12	12
11	Passed!	find(s, *args)	['the letter after p will not be found in this string', 'q']	-1	-1
12	Failed!	gcd(a,b)	[12, 18]	6	18
13	Passed!	ljust(s, width, *args)	['test', 7]	test	test
14	Passed!	atoi(s,base=10)	['00101100', 2]	44	44
15	Passed!	atoi(s,base=10)	['1BC', 16]	444	444
16	Passed!	expandtabs(s, tabsize=8)	['hello\tworld', 1]	hello world	hello world
17	Passed!	ljust(s, width, *args)	['test', 7]	test	test
18	Failed!	zfill(x, width)	[123, 8]	00000123	0000000000000123
19	Passed!	center(s, width, *args)	['test', 10]	test	test
20	Passed!	wrap(text, width=70, **kwargs)	['hello world and all who inhabit it', 8]	['hello', 'world', 'and all', 'who', 'inhabit', 'it']	['hello', 'world', 'and all', 'who', 'inhabit', 'it']
21	Failed!	unquote(s)	['abc def']	abc def	abcPf
22	Passed!	wrap(text, width=70, **kwargs)	['hello world and all who inhabit it', 15]	['hello world and', 'all who inhabit', 'it']	['hello world and', 'all who inhabit', 'it']
23	Passed!	zfill(x, width)	[123, 0]	123	123
24	Passed!	rfind(s, *args)	['looking for the last instance of the letter g', 'g']	44	44
25	Passed!	replace(s, old, new)	['how will it handle a t on of useless space ?', ' ', '_']	how_will___it_handle___a___t_on__of_useless_space___?	how_will___it_handle___a___t_on__of_useless_space___?

Figure 4. Example output from the Testing framework, with faults injected

Results

Seen in figure 4, the Framework produces and opens and subsequently displays a listing of all the test cases performed in HTML. This listing includes including their number, status, input, expected output, and the function being tested. The framework is able to continue testing even when errors are found, and marks the test as failing.

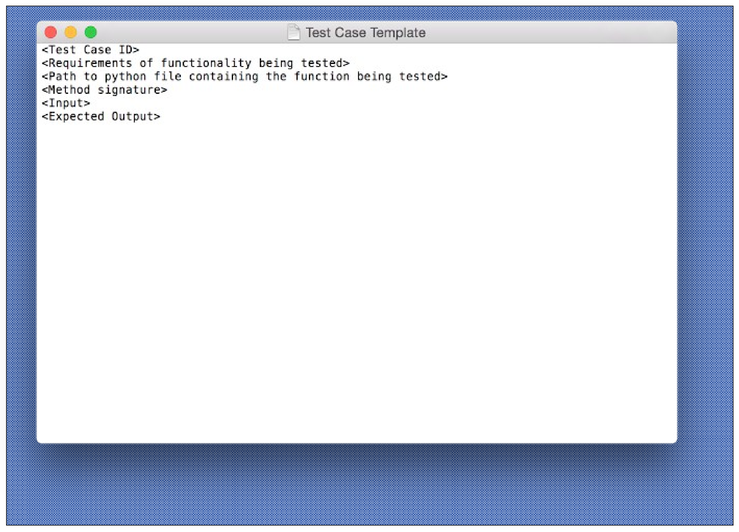

Figure 5. Test case template used by our Automated Testing Framework

Conclusions

The current Testing Framework is functioning as intended within the scope of the planned suite of 25 test cases. Additionally it has the added benefit of the test cases, as well as the script making no explicit references to the Jython project specifically, only its libraries. As such, the framework could be used to test any project written in python which contains standalone functions

Going forward the framework has a couple areas of note which could be improved upon to extend it's capabilities, beyond the scope of this project. Primarily with regards to its limitation of only being able to test static and standalone functions, incorporating the ability to test functions within classes could be a major improvement. Additionally, some way to run tests without keeping them all in memory at one time could be invaluable in processing much larger test suites