# CS 562 Final Report On Project

Student: Weiyu Lin

ID: 932-479-491

Mail: Linweiy@oregonstate.edu

---

## My accomplishment

- Short introduction

The python library I have tested is called "algorithms" on github. It is a python library about data structure and classic algorithms(searching, sorting etc.). The library can be cloned from this Link. A quick install method to this library is typing following into terminal:

```
    $ pip install algorithms
```

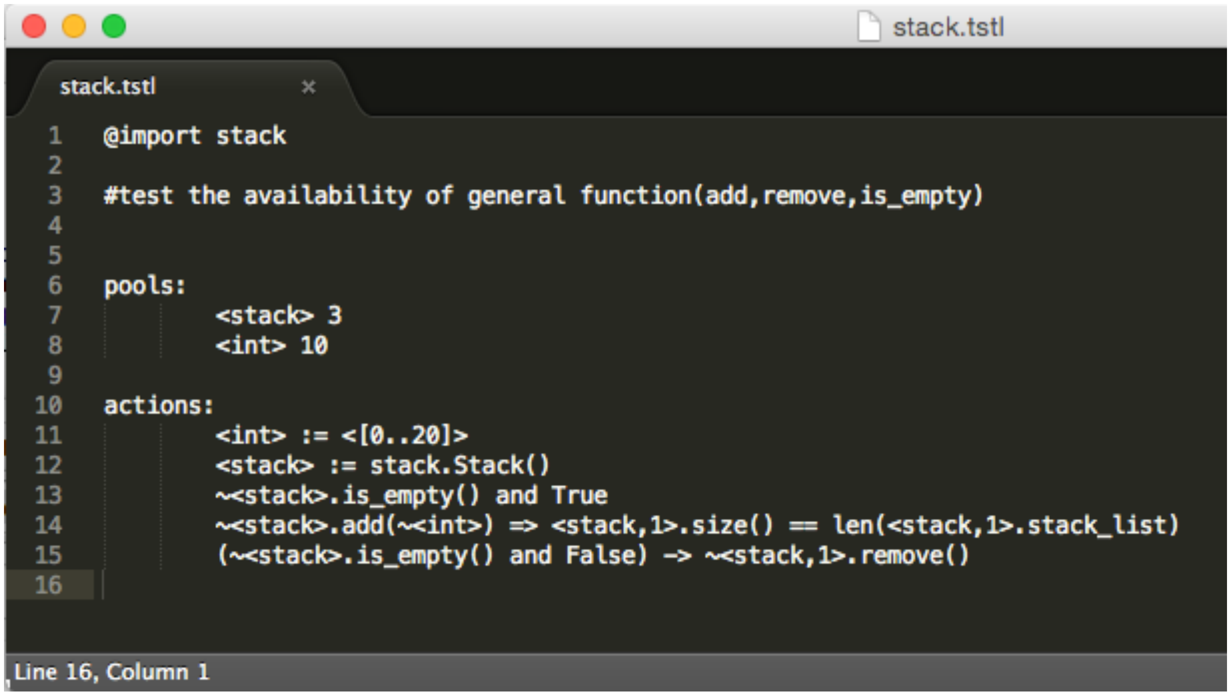
**[Note]**:For the convenient issue, I have copy and paste it down and attached it under my file on Github.

- My work

I have set proposal goals that I'll test data structures and searching algorithms from this library. Since the seaching algorithms are far more difference as I thought when I set the goal, I have adjust my goal to focus on data structure insteading of focusing both. I have tested `stack.py`, `queue.py`, `SingleLinkedList`, `binary_search_tree.py` from the library mentioned above by TSTL.

- Data structure

for the section, I tried to check the correctness of data structure by passing one value into specific data structure to see if getting corresponding answer. I Checked whether all the functions are work well I give an Stack as simple example here.



```
@import stack

#test the availability of general function(add,remove,is_empty)


pools:
        <stack> 3
        <int> 10

actions:
        <int> := <[0..20]>
        <stack> := stack.Stack()
        ~<stack>.is_empty() and True
        ~<stack>.add(~<int>) => <stack,1>.size() == len(<stack,1>.stack_list)
        (~<stack>.is_empty() and False) -> ~<stack,1>.remove()
```

`stack.py` has four basic(essential) function,which are `stack.add()`, `stack.remove`, `stack.isempty()`, `stack.size()`. For my TSTL file, I wrote script to make sure all the functions of stack are covered.

```
actions:
<int> := <[0..20]>
<stack> := stack.Stack()
~<stack>.is_empty() and True
~<stack>.add(~<int>) => <stack,1>.size() == len(<stack,1>.stack_list)
(~<stack>.is_empty() and False) -> ~<stack,1>.remove()
```

This script runs well under `randomtester.py`, which means all the function works well when implemented.

```
<stack> := stack.Stack()
~<stack>.is_empty() and True
```

This script works for check if stack is empty when it just after initialization.

```
~<stack>.add(~<int>) => <stack,1>.size() == len(<stack,1>.stack_list)
(~<stack>.is_empty() and False) -> ~<stack,1>.remove()
```

This script works for check if the size number of stack meets the expection when add values into stack.

```
    (~<stack>.is_empty() and False) -> ~<stack,1>.remove()
```

The last code move values away from stack after checking the status of `is_empty()` is false.

This example shows my thought on how to testing by TSTL. for the `queue.py`, `binary_search_tree.py` parts, I use very similar thought to write the TSTL script testing. These TSTL files can be found at CS562w16/Project/Linweiy.

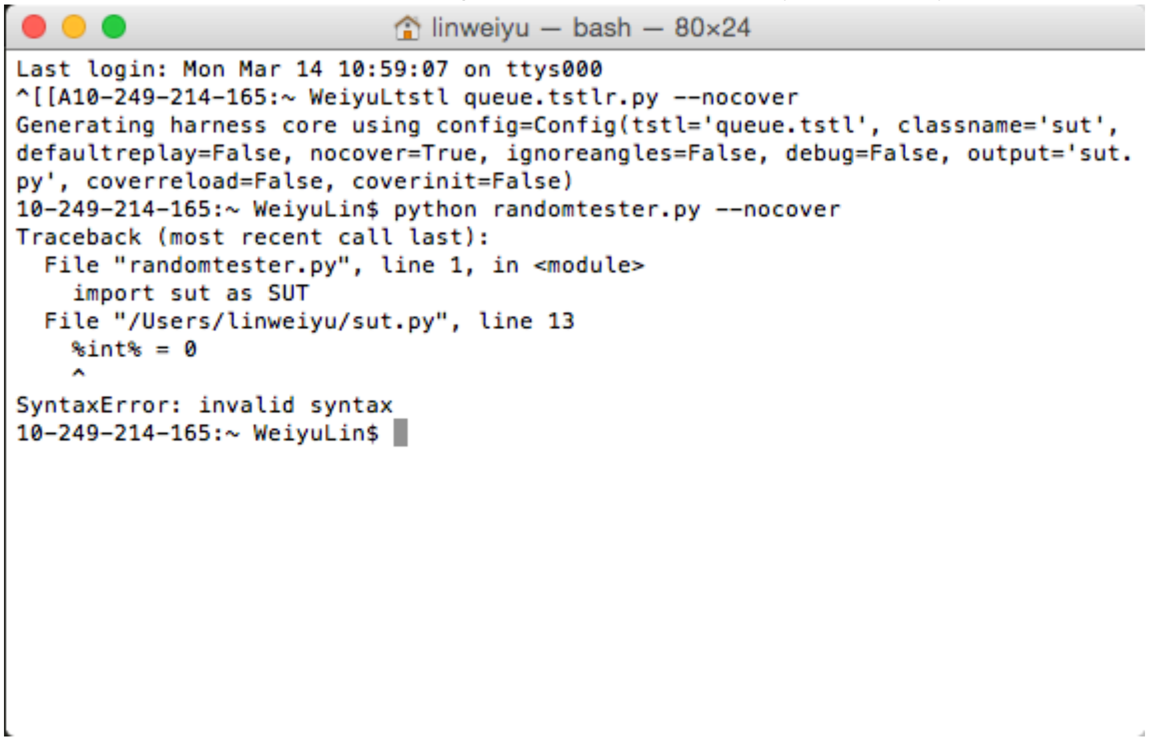
## Did you find bugs?

The object library is widely used with more than 1k Star. As you can see in "My accomplishment" part, I have tried to write test script to cover every function in each test object. Sad to say,  though I tried very hard to see if I can get any kind of bugs in the library I test for, no bug was found until I edit this report.

## How well does the tester work?

The `randomtester.py` for TSTL designs good.

- helpful when got failed The `randomtester.py` provides very helpful explanation by pointing out the specific line and reason when I got fail, it help me correct my code quickly.



```
Last login: Mon Mar 14 10:59:07 on ttys000
^[[A10-249-214-165:~ WeiyuLtstl queue.tstlr.py --nocover
Generating harness core using config=Config(tstl='queue.tstl', classname='sut',
defaultreplay=False, nocover=True, ignoreangles=False, debug=False, output='sut.
py', coverreload=False, coverinit=False)
10-249-214-165:~ WeiyuLin$ python randomtester.py --nocover
Traceback (most recent call last):
  File "randomtester.py", line 1, in <module>
    import sut as SUT
  File "/Users/linweiyu/sut.py", line 13
    %int% = 0
    ^
SyntaxError: invalid syntax
10-249-214-165:~ WeiyuLin$ 
```

- Perform good to test humongous cases As we know, `randomtester.py` is random test for `SUT.py`. In order to dig bugs, `randomtester.py` sometimes required to test humongous cases generated by the variable range setting in TSTL file. `randomtester.py` performs really wonderful for this situation.

## What is wrong with TSTL?

- More test cases

The example `AVLTree.py` is very helpful for fresh like me. I believe people can get more deep understanding if more examples can be provided. Also, varities of test examples can be provide, it is really help for people who use TSTL testing for network module or other library if relevant examples are given.

- Support for more programming languages

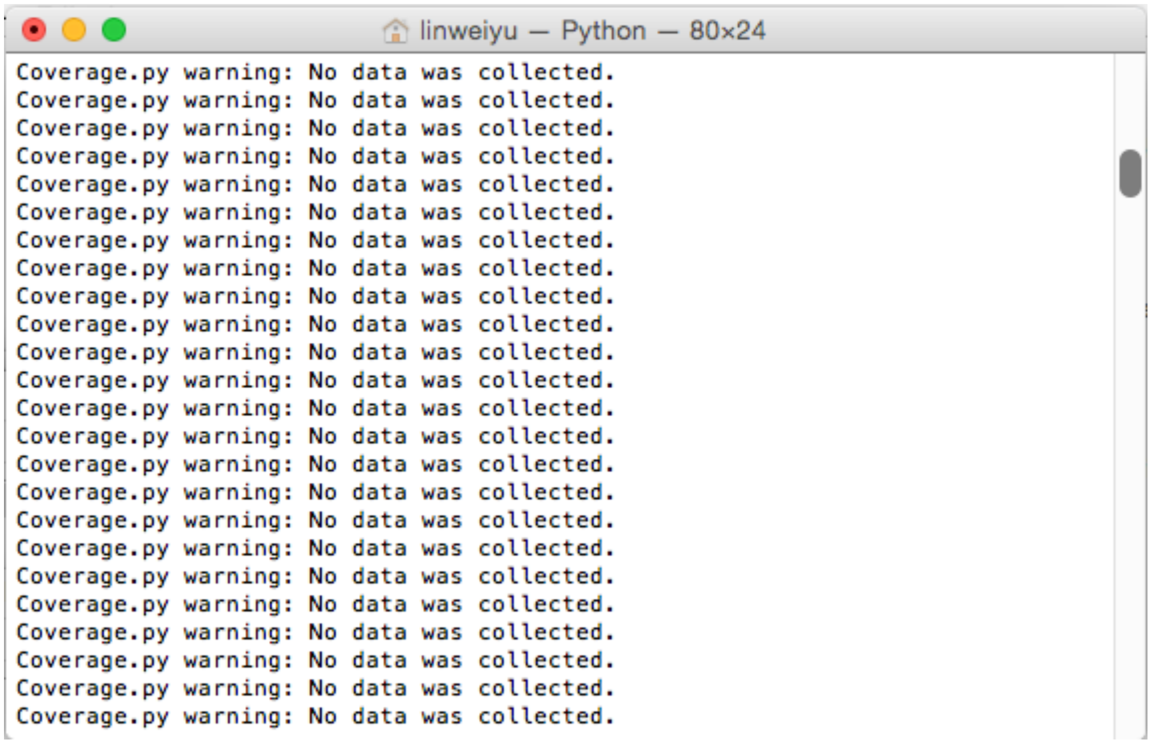
TSTL is only support for `Python` now. Though `Python` is a very popular language, TSTL can be wilely used if it supports for more programming language (`Java`, `C`, `C++` etc.)

- A tutorial for TSTL

It is awesome if TSTL has a tutorial to introduce its Syntax, Feature. Fresh can be start quickly with help of tutorial.

- Robustness can be improived

Some unexpected warn still existence when running tests. For example, sometimes terminal will give a `No data was collected.` warning.



```
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
```

## What is good about TSTL?

- Easy to start for fresh

TSTL is easy to start for fresh like me who have little testing experience. The prerequisites for a starter is nothing but able to programming by `Python`.

- Automatic input

One of evident advantage of TSTL is you can set rangealbe variables as input to testing your object. This excellent feature save both engery and time.

- Clear Syntax

The syntax for TSTL is very clear and easy to understand. for example, `pool` section state variable and `action` is the script for implementation, and tester can also define own function in `<@.....@>`, which make TSTL easy to review.

- SUT file

I believe SUT file is the core advantage of TSTL. Tester do not need to write bunches of harness by insteading of SUT file, which makes TSTL efficient and convenient.

## Coverage summaries, any other important data on

## the test effort

```
    Name                           Stmts    Miss  Cover   Missing

    "------------------------------------------------------------------------"

    stack.py                       12       5     58%   17, 25, 34, 42, 50

    queue.py                       13       5     62%   21, 29, 38, 46, 54


    singly_linked_list.py          45      33     27%   17-18, 21, 24, 27, 30, 36-37,
44-47, 54-71, 78-87, 93

    binary_search_tree.py        169     137     19%   20-24, 33, 36-39, 49, 59, 62-70, 80,
90, 95-109, 119, 125-133, 143-147, 153-161, 171-175, 182-199, 210-214, 221-236, 247-
251, 257-266, 276-280, 283-292, 302, 305-323, 333, 336-341, 352, 355-360, 370, 373-
384, 394-395
```

**[Note]**: The terminal output file of 4 data structures can be found at [CS562w16/Project/Linweiy](CS562w16/Project/Linweiy).

## Summary

As a starter to testing. I am very glad to learn TSTL on CS562 this term. This is
In this class I am very happy to learn the TSTL language. although there are still some drawbacks exists (for example,"No data was collected" problem ), I think this is a very useful and convenient language in testing area. Especially the good performance on implementing large numbers of test data in `randomtester.py` which states by TSTL script. In the meanwhile, I have reinforce the knowledge learned from CS261 and CS325. I have strength both testing skills, python skills, the understanding of data strucutures.

word count: 1150