

Part 2: Unit testing activity

For this task, I chose the python language and statistical program snakestats. In course of this work, I created three sets of tests for different functions that are used to make the Student's t-test.

The first target function I tried to create using test-driven development is the arithmetic mean. The arithmetic mean (or average) is a simple and common function used as a cornerstone for more advanced statistical measures. The first test I wrote is simple - we try to calculate the average of a set of numbers.

TESTS.Py	SNAKESTATS.Py
<pre>Sem3 > SoftwareDesign > Midterm > tests.py > ... 1 import unittest 2 import snakestats 3 4 5 class StatisticsTests(unittest.TestCase): 6 def test_11_Average(self): 7 avg = snakestats.average([2, 3, 4]) 8 self.assertEqual(avg, 3) 9 10 11 unittest.main(argv=['ignored', '-v'], exit=False)</pre>	<pre>Sem3 > SoftwareDesign > Midterm > snakestats.py 1 </pre>
<pre>test_11_Average (__main__.StatisticsTests) ... ERROR ===== ERROR: test_11_Average (__main__.StatisticsTests) ----- Traceback (most recent call last): File "C:\Study\Sem3\SoftwareDesign\Midterm\tests.py", line 7, in test_11_Average avg = snakestats.average([2, 3, 4]) AttributeError: module 'snakestats' has no attribute 'average' ----- Ran 1 test in 0.001s FAILED (errors=1)</pre>	

Because the “snakestats” file is empty, the test fails, so now I add some code (minimum required to pass the failed test).

<pre>Sem3 > SoftwareDesign > Midterm > tests.py > ... 1 import unittest 2 import snakestats 3 4 5 class StatisticsTests(unittest.TestCase): 6 def test_11_Average(self): 7 avg = snakestats.average([2, 3, 4]) 8 self.assertEqual(avg, 3) 9 10 11 unittest.main(argv=['ignored', '-v'], exit=False)</pre>	<pre>Sem3 > SoftwareDesign > Midterm > snakestats.py > ... 1 from logging import exception, log 2 import math 3 4 5 def average(arg): 6 return 3</pre>
<pre>test_11_Average (__main__.StatisticsTests) ... ok ----- Ran 1 test in 0.000s OK</pre>	

The first test passed successfully, so I create the next one. I decided to try and increase the argument

size and values and test again. Because our function only returns a specific integer, it obviously fails.

```
Sem3 > SoftwareDesign > Midterm > tests.py > ...
1 import unittest
2 import snakestats
3
4
5 class StatisticsTests(unittest.TestCase):
6     def test_11_Average(self):
7         avg = snakestats.average([2, 3, 4])
8         self.assertEqual(avg, 3)
9
10    def test_12_Average(self):
11        avg = snakestats.average([2, 3, 4, 5, 6])
12        self.assertEqual(avg, 4)
13
14
15 unittest.main(argv=['ignored', '-v'], exit=False)
```

```
Sem3 > SoftwareDesign > Midterm > snakestats.py > ...
1 from logging import exception, log
2 import math
3
4
5 def average(arg):
6     return 3
```

```
test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... FAIL

=====
FAIL: test_12_Average (__main__.StatisticsTests)
-----
Traceback (most recent call last):
  File "C:\Study\Sem3\SoftwareDesign\Midterm\tests.py", line 12, in test_12_Average
    self.assertEqual(avg, 4)
AssertionError: 3 != 4

-----
Ran 2 tests in 0.001s

FAILED (failures=1)
```

To pass this and other tests, we need to modify it to return the result based on the argument passed to the function.

```
Sem3 > SoftwareDesign > Midterm > tests.py > ...
1 import unittest
2 import snakestats
3
4
5 class StatisticsTests(unittest.TestCase):
6     def test_11_Average(self):
7         avg = snakestats.average([2, 3, 4])
8         self.assertEqual(avg, 3)
9
10    def test_12_Average(self):
11        avg = snakestats.average([2, 3, 4, 5, 6])
12        self.assertEqual(avg, 4)
13
14
15 unittest.main(argv=['ignored', '-v'], exit=False)
```

```
Sem3 > SoftwareDesign > Midterm > snakestats.py > ...
1 from logging import exception, log
2 import math
3
4
5 def average(arg):
6     sum = 0
7     n = len(arg)
8     for i in range(0, n):
9         sum += arg[i]
10    return sum/n
```

```
test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... ok

-----
Ran 2 tests in 0.001s

OK
```

Now, for the final test, I tried passing the wrong argument to push the function into an exception throw.

```
Sem3 > SoftwareDesign > Midterm > tests.py > ...
1 import unittest
2 import snakestats
3
4
5 class StatisticsTests(unittest.TestCase):
6 > def test_11_Average(self): ...
9
10 > def test_12_Average(self): ...
13
14     def test_13_Average(self):
15         avg = snakestats.average([2, 'b', 18, 20])
16         self.assertRaises(Exception)
17
18
19 unittest.main(argv=['ignored', '-v'], exit=False)
```

```
Sem3 > SoftwareDesign > Midterm > snakestats.py > ...
1 from logging import exception, log
2 import math
3
4
5 def average(arg):
6     sum = 0
7     n = len(arg)
8     for i in range(0, n):
9         sum += arg[i]
10    return sum/n
```

```
test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... ok
test_13_Average (__main__.StatisticsTests) ... ERROR

=====
ERROR: test_13_Average (__main__.StatisticsTests)
-----
Traceback (most recent call last):
  File "C:\Study\Sem3\SoftwareDesign\Midterm\tests.py", line 15, in test_13_Average
    avg = snakestats.average([2, 'b', 18, 20])
  File "C:\Study\Sem3\SoftwareDesign\Midterm\snakestats.py", line 9, in average
    sum += arg[i]
TypeError: unsupported operand type(s) for +=: 'int' and 'str'

-----
Ran 3 tests in 0.002s

FAILED (errors=1)
```

To pass the final test, the function is equipped with the "try" statement, that handles errors associated with arguments (wrong data type, empty array, etc.). And now, the test is successfully passed because the function throws out an exception for the bad input.

```
Sem3 > SoftwareDesign > Midterm > tests.py > ...
1 import unittest
2 import snakestats
3
4
5 class StatisticsTests(unittest.TestCase):
6 > def test_11_Average(self): ...
9
10 > def test_12_Average(self): ...
13
14     def test_13_Average(self):
15         avg = snakestats.average([2, 'b', 18, 20])
16         self.assertRaises(Exception)
17
18
19 unittest.main(argv=['ignored', '-v'], exit=False)
```

```
Sem3 > SoftwareDesign > Midterm > snakestats.py > ...
1 from logging import exception, log
2 import math
3
4
5 def average(arg):
6     try:
7         if(type(arg) == list):
8             sum = 0
9             n = len(arg)
10            for i in range(0, n):
11                sum += arg[i]
12            return sum/n
13        else:
14            return arg
15    except Exception as Argument:
16        return Argument
```

```
test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... ok
test_13_Average (__main__.StatisticsTests) ... ok

-----
Ran 3 tests in 0.001s

OK
```

The next set of tests is designed for the "variance" function, which is used to calculate the standard deviation. To calculate variance, we calculate the average sum up the squares on the differences of each number from the average and then divide it by the length of the list. The first test is to calculate the variance of the simple list. As expected, without the code, this test fails.

<pre> 5 class StatisticsTests(unittest.TestCase): 6 > def test_11_Average(self): ... 9 10 > def test_12_Average(self): ... 13 14 > def test_13_Average(self): ... 17 18 def test_21_Variance(self): 19 var = snakestats.variance([2, 2, 3, 5]) 20 self.assertEqual(var, 2) 21 self.assertGreaterEqual(var, 0) 22 23 24 unittest.main(argv=['ignored', '-v'], exit=False) </pre>	<pre> Sem3 > SoftwareDesign > Midterm > snakestats.py > ... 1 from logging import exception, log 2 import math 3 4 5 > def average(arg): ... 17 </pre>
<pre> test_11_Average (__main__.StatisticsTests) ... ok test_12_Average (__main__.StatisticsTests) ... ok test_13_Average (__main__.StatisticsTests) ... ok test_21_Variance (__main__.StatisticsTests) ... ERROR ===== ERROR: test_21_Variance (__main__.StatisticsTests) ----- Traceback (most recent call last): File "C:\Study\Sem3\SoftwareDesign\Midterm\tests.py", line 19, in test_21_Variance var = snakestats.variance([2, 2, 3, 5]) AttributeError: module 'snakestats' has no attribute 'variance' ----- Ran 4 tests in 0.002s FAILED (errors=1) </pre>	

In the same way, our first iteration of the function "variance" returns the value for the tested input without any calculations.

<pre> 5 class StatisticsTests(unittest.TestCase): 6 > def test_11_Average(self): ... 9 10 > def test_12_Average(self): ... 13 14 > def test_13_Average(self): ... 17 18 def test_21_Variance(self): 19 var = snakestats.variance([2, 2, 3, 5]) 20 self.assertEqual(var, 2) 21 self.assertGreaterEqual(var, 0) 22 23 24 unittest.main(argv=['ignored', '-v'], exit=False) </pre>	<pre> Sem3 > SoftwareDesign > Midterm > snakestats.py > ... 1 from logging import exception, log 2 import math 3 4 5 > def average(arg): ... 17 18 19 def variance(arg): 20 return 2 </pre>
<pre> test_11_Average (__main__.StatisticsTests) ... ok test_12_Average (__main__.StatisticsTests) ... ok test_13_Average (__main__.StatisticsTests) ... ok test_21_Variance (__main__.StatisticsTests) ... ok ----- Ran 4 tests in 0.001s OK </pre>	

But if we try another input, the function returns with an error, as expected.

```
5 class StatisticsTests(unittest.TestCase):
6 > def test_11_Average(self): ...
9
10 > def test_12_Average(self): ...
13
14 > def test_13_Average(self): ...
17
18 > def test_21_Variance(self): ...
22
23 def test_22_Variance(self):
24     var = snakestats.variance([14, 12, 22, 23, 16,
25                               24, 22, 20, 18])
26     self.assertEqual(var, 18)
27     self.assertGreaterEqual(var, 0)
28
29 unittest.main(argv=['ignored', '-v'], exit=False)
```

```
Sem3 > SoftwareDesign > Midterm > snakestats.py > ...
1 from logging import exception, log
2 import math
3
4
5 > def average(arg): ...
17
18
19 def variance(arg):
20     return 2
```

```
test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... ok
test_13_Average (__main__.StatisticsTests) ... ok
test_21_Variance (__main__.StatisticsTests) ... ok
test_22_Variance (__main__.StatisticsTests) ... FAIL

=====
FAIL: test_22_Variance (__main__.StatisticsTests)
-----
Traceback (most recent call last):
  File "C:\Study\Sem3\SoftwareDesign\Midterm\tests.py", line 26, in test_22_Variance
    self.assertEqual(var, 18)
AssertionError: 2 != 18

-----
Ran 5 tests in 0.002s

FAILED (failures=1)
```

To make our function applicable for any correct input, we modify the function to consider argument values.

```
5 class StatisticsTests(unittest.TestCase):
6 > def test_11_Average(self): ...
9
10 > def test_12_Average(self): ...
13
14 > def test_13_Average(self): ...
17
18 > def test_21_Variance(self): ...
22
23 def test_22_Variance(self):
24     var = snakestats.variance([14, 12, 22, 23, 16,
25                               24, 22, 20, 18])
26     self.assertEqual(var, 18)
27     self.assertGreaterEqual(var, 0)
28
29 unittest.main(argv=['ignored', '-v'], exit=False)
```

```
1 from logging import exception, log
2 import math
3
4
5 > def average(arg): ...
17
18 def variance(arg):
19     n = len(arg)
20     avr = average(arg)
21     sq_sum = 0
22     for el in arg:
23         sq_sum += (el-avr) * (el - avr)
24     return sq_sum/(n-1)
```

```
test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... ok
test_13_Average (__main__.StatisticsTests) ... ok
test_21_Variance (__main__.StatisticsTests) ... ok
test_22_Variance (__main__.StatisticsTests) ... ok

-----
Ran 5 tests in 0.001s

OK
```

This way, the function will return the correct variance if the user inputs input with numbers, but once again, this function fails for incorrect argument.

```
5 class StatisticsTests(unittest.TestCase):
6 > def test_11_Average(self): ...
9
10 > def test_12_Average(self): ...
13
14 > def test_13_Average(self): ...
17
18 > def test_21_Variance(self): ...
22
23 > def test_22_Variance(self): ...
28
29 def test_23_Variance(self):
30     var = snakestats.variance(10)
31     self.assertRaises(Exception)
32
33
34 unittest.main(argv=['ignored', '-v'], exit=False)
```

```
1 from logging import exception, log
2 import math
3
4
5 > def average(arg): ...
17
18 def variance(arg):
19     n = len(arg)
20     avr = average(arg)
21     sq_sum = 0
22     for el in arg:
23         sq_sum += (el-avr) * (el - avr)
24     return sq_sum/(n-1)
```

```
test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... ok
test_13_Average (__main__.StatisticsTests) ... ok
test_21_Variance (__main__.StatisticsTests) ... ok
test_22_Variance (__main__.StatisticsTests) ... ok
test_23_Variance (__main__.StatisticsTests) ... ERROR

=====
ERROR: test_23_Variance (__main__.StatisticsTests)
-----
Traceback (most recent call last):
  File "C:\Study\Sem3\SoftwareDesign\Midterm\tests.py", line 30, in test_23_Variance
    var = snakestats.variance(10)
  File "C:\Study\Sem3\SoftwareDesign\Midterm\snakestats.py", line 20, in variance
    n = len(arg)
TypeError: object of type 'int' has no len()

-----
Ran 6 tests in 0.002s

FAILED (errors=1)
```

To fix this, we must check for errors and throw exceptions correctly.

```
5 class StatisticsTests(unittest.TestCase):
6 > def test_11_Average(self): ...
9
10 > def test_12_Average(self): ...
13
14 > def test_13_Average(self): ...
17
18 > def test_21_Variance(self): ...
22
23 > def test_22_Variance(self): ...
28
29 def test_23_Variance(self):
30     var = snakestats.variance(10)
31     self.assertRaises(Exception)
32
33
34 unittest.main(argv=['ignored', '-v'], exit=False)
```

```
19 def variance(arg):
20     try:
21         n = len(arg)
22         avr = average(arg)
23         sq_sum = 0
24         for el in arg:
25             sq_sum += (el-avr) * (el - avr)
26         return sq_sum/(n-1)
27     except Exception as Argument:
28         return Argument
```

```
test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... ok
test_13_Average (__main__.StatisticsTests) ... ok
test_21_Variance (__main__.StatisticsTests) ... ok
test_22_Variance (__main__.StatisticsTests) ... ok
test_23_Variance (__main__.StatisticsTests) ... ok

-----
Ran 6 tests in 0.002s

OK
```

And for the last target function, we pick a "statistics analyzer" that returns True if the correlation is significant. This function shall take the two data sets and critical value, calculate the t-statistic and check it against the t criteria. The first test fails because there is no function yet.

```
33 def test_31_Significance(self):
34     sig = snakestats.significance([1, 4, 6, 7],
35                                   [3, 6, 6, 8], 2.447)
36     self.assertTrue(sig)
37
38
39 unittest.main(argv=['ignored', '-v'], exit=False)
```

```
1  from logging import exception, log
2  import math
3
4
5 > def average(arg): ...
17
18
19 > def variance(arg): ...
29
30
```

```
test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... ok
test_13_Average (__main__.StatisticsTests) ... ok
test_21_Variance (__main__.StatisticsTests) ... ok
test_22_Variance (__main__.StatisticsTests) ... ok
test_23_Variance (__main__.StatisticsTests) ... ok
test_31_Significance (__main__.StatisticsTests) ... ERROR

=====
ERROR: test_31_Significance (__main__.StatisticsTests)
-----
Traceback (most recent call last):
  File "C:\Study\Sem3\SoftwareDesign\Midterm\tests.py", line 34, in test_31_Significance
    sig = snakestats.significance([1, 4, 6, 7],
AttributeError: module 'snakestats' has no attribute 'significance'

-----
Ran 7 tests in 0.002s

FAILED (errors=1)
```

We created the function with one return statement and passed the first test.

```
33 def test_31_Significance(self):
34     sig = snakestats.significance([1, 4, 6, 7],
35                                   [3, 6, 6, 8], 2.447)
36     self.assertTrue(sig)
37
38
39 unittest.main(argv=['ignored', '-v'], exit=False)
```

```
31 def significance(arg1, arg2, arg3):
32     return True
```

```
test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... ok
test_13_Average (__main__.StatisticsTests) ... ok
test_21_Variance (__main__.StatisticsTests) ... ok
test_22_Variance (__main__.StatisticsTests) ... ok
test_23_Variance (__main__.StatisticsTests) ... ok
test_31_Significance (__main__.StatisticsTests) ... ok

-----
Ran 7 tests in 0.003s

OK
```

But because there is no formula, the second test fails.


```

38     def test_32_Significance(self):
39         sig = snakestats.significance([1, 2, 3, 4, 5, 6, 7, 8, 9],
40                                     [8, 13, 3, 9, 7, 12, 8, 13, 11],
41                                     2.120)
42         self.assertLogs()
43         self.assertFalse(sig)
44
45
46 unittest.main(argv=['ignored', '-v'], exit=False)

```

```

31 def significance(arr1, arr2, arg3):
32     return True

```

```

test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... ok
test_13_Average (__main__.StatisticsTests) ... ok
test_21_Variance (__main__.StatisticsTests) ... ok
test_22_Variance (__main__.StatisticsTests) ... ok
test_23_Variance (__main__.StatisticsTests) ... ok
test_31_Significance (__main__.StatisticsTests) ... ok
test_32_Significance (__main__.StatisticsTests) ... FAIL

=====
FAIL: test_32_Significance (__main__.StatisticsTests)
-----
Traceback (most recent call last):
  File "C:\Study\Sem3\SoftwareDesign\Midterm\tests.py", line 43, in test_32_Significance
    self.assertFalse(sig)
AssertionError: True is not false

-----
Ran 8 tests in 0.003s

FAILED (failures=1)

```

Now we add the correct formula, and the second test passes successfully.

```

38     def test_32_Significance(self):
39         sig = snakestats.significance([1, 2, 3, 4, 5, 6, 7, 8, 9],
40                                     [8, 13, 3, 9, 7, 12, 8, 13, 11],
41                                     2.120)
42         self.assertLogs()
43         self.assertFalse(sig)
44
45
46 unittest.main(argv=['ignored', '-v'], exit=False)

```

```

31 def significance(arr1, arr2, criticalT):
32     n1 = len(arr1)
33     n2 = len(arr2)
34     avg1 = average(arr1)
35     avg2 = average(arr2)
36     var1 = variance(arr1)
37     var2 = variance(arr2)
38
39     t = (max(avg1, avg2)-min(avg1, avg2)) / math.sqrt(var1/n1 + var2/n2)
40
41     return t < criticalT

```

```

test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... ok
test_13_Average (__main__.StatisticsTests) ... ok
test_21_Variance (__main__.StatisticsTests) ... ok
test_22_Variance (__main__.StatisticsTests) ... ok
test_23_Variance (__main__.StatisticsTests) ... ok
test_31_Significance (__main__.StatisticsTests) ... ok
test_32_Significance (__main__.StatisticsTests) ... ok

-----
Ran 8 tests in 0.003s

OK

```

For the last test, we again throw the wrong argument.

```

45     def test_33_Significance(self):
46         sig = snakestats.significance('Hello, World!', 44, 12.706)
47         self.assertRaises(Exception)
48
49
50 unittest.main(argv=['ignored', '-v'], exit=False)

```

```

31 def significance(arr1, arr2, criticalT):
32     n1 = len(arr1)
33     n2 = len(arr2)
34     avg1 = average(arr1)
35     avg2 = average(arr2)
36     var1 = variance(arr1)
37     var2 = variance(arr2)
38
39     t = (max(avg1, avg2)-min(avg1, avg2)) / math.sqrt(var1/n1 + var2/n2)
40
41     return t < criticalT

```



```

test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... ok
test_13_Average (__main__.StatisticsTests) ... ok
test_21_Variance (__main__.StatisticsTests) ... ok
test_22_Variance (__main__.StatisticsTests) ... ok
test_23_Variance (__main__.StatisticsTests) ... ok
test_31_Significance (__main__.StatisticsTests) ... ok
test_32_Significance (__main__.StatisticsTests) ... ok
test_33_Significance (__main__.StatisticsTests) ... ERROR

=====
ERROR: test_33_Significance (__main__.StatisticsTests)
-----
Traceback (most recent call last):
  File "C:\Study\Sem3\SoftwareDesign\Midterm\tests.py", line 46, in test_33_Significance
    sig = snakestats.significance('Hello, World!', 44, 12.706)
  File "C:\Study\Sem3\SoftwareDesign\Midterm\snakestats.py", line 33, in significance
    n2 = len(arr2)
TypeError: object of type 'int' has no len()

-----
Ran 9 tests in 0.004s

FAILED (errors=1)

```

To handle argument problems, we add an exception handler, and finally, pass all tests successfully.

```

45     def test_33_Significance(self):
46         sig = snakestats.significance('Hello, World!', 44, 12.706)
47         self.assertRaises(Exception)
48
49
50 unittest.main(argv=['ignored', '-v'], exit=False)

```

```

31 def significance(arr1, arr2, criticalT):
32     try:
33         n1 = len(arr1)
34         n2 = len(arr2)
35         avg1 = average(arr1)
36         avg2 = average(arr2)
37         var1 = variance(arr1)
38         var2 = variance(arr2)
39
40         t = (max(avg1, avg2) - min(avg1, avg2)) / math.sqrt(var1/n1 + var2/n2)
41
42         return t < criticalT
43     except Exception as Argument:
44         return Argument

```

```

test_11_Average (__main__.StatisticsTests) ... ok
test_12_Average (__main__.StatisticsTests) ... ok
test_13_Average (__main__.StatisticsTests) ... ok
test_21_Variance (__main__.StatisticsTests) ... ok
test_22_Variance (__main__.StatisticsTests) ... ok
test_23_Variance (__main__.StatisticsTests) ... ok
test_31_Significance (__main__.StatisticsTests) ... ok
test_32_Significance (__main__.StatisticsTests) ... ok
test_33_Significance (__main__.StatisticsTests) ... ok

-----
Ran 9 tests in 0.030s

OK

```