UNIVERSITY OF LONDON

Name – Rishi Sankhla
Student No. -200102874

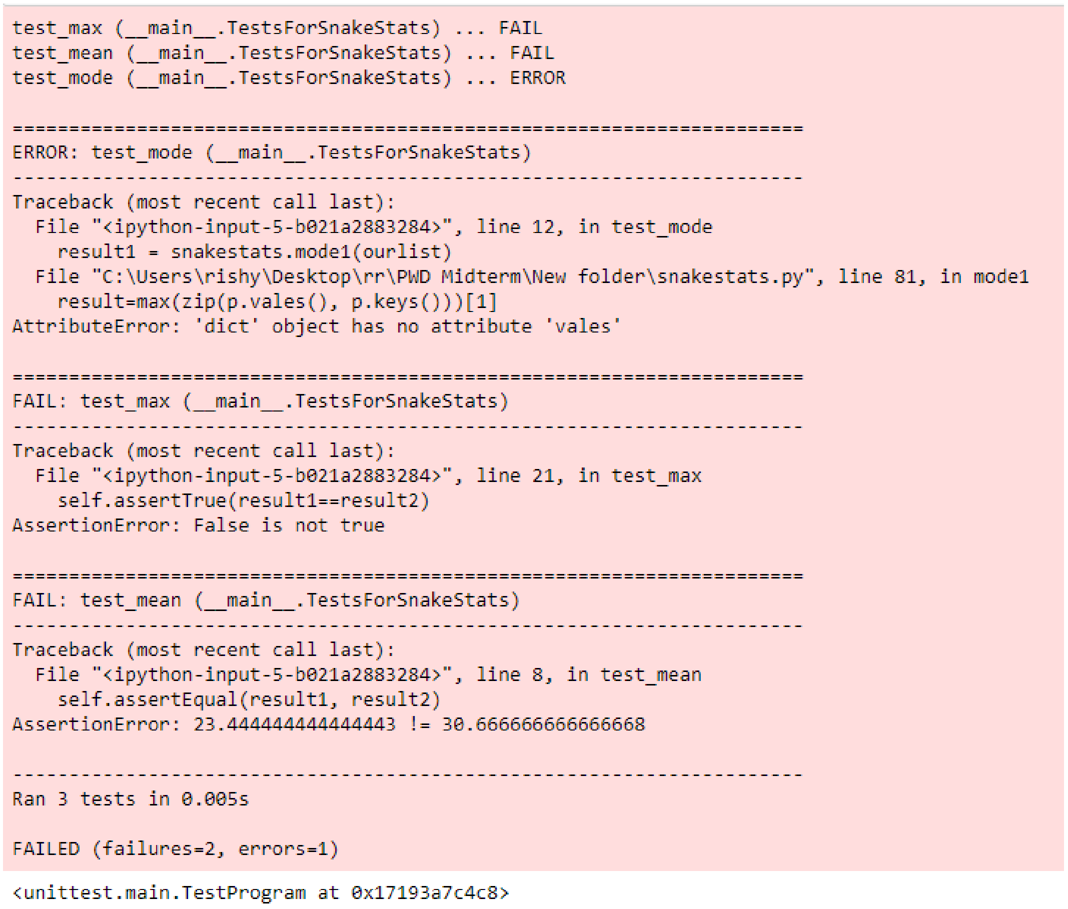
Mid-term Coursework 1 submission [001]
Cm-2010-Software Design and Development
Midterm Question paper Version: 20201008
Part 2: Unit testing activity

Test 1(set-1): this test is to calculate the mean of list, here we are comparing our algorithm output with the statistic library of python using assert-equal function the first test was failed because it can calculate mean only for list of size 4. Whereas in second algorithm we do the summation of whole list and then divide it by its length.

Test 2(set-1): this test is to calculate the mode of list, then compare it with statistic library output using assert-true function, here my approach was to build a frequency table of each element using dictionary object and then extracts the highest frequency element, the first test was failed because of dictionary attribute-error (happened because of spelling mistake).

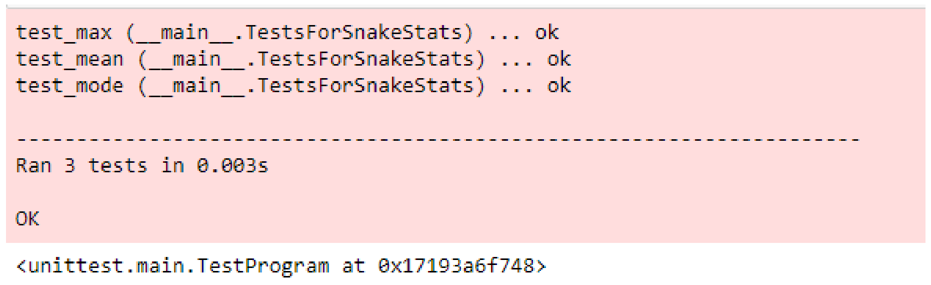
Test 3(set-1): this test is to find the maximum number in our list, then compare it with numpy library output using assert-true function, here the first approach is to compare numbers with each other and keeping the record of highest element but it was not scalable therefore in second approach we iterate over each element and compare it with our maximum stored so far.

Set 1 Failed Screenshot:



```
test_max (__main__.TestsForSnakeStats) ... FAIL
test_mean (__main__.TestsForSnakeStats) ... FAIL
test_mode (__main__.TestsForSnakeStats) ... ERROR

======================================================================
ERROR: test_mode (__main__.TestsForSnakeStats)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-5-b021a2883284>", line 12, in test_mode
    result1 = snakestats.mode1(ourlist)
  File "C:\Users\rishy\Desktop\rr\PWD Midterm\New folder\snakestats.py", line 81, in mode1
    result=max(zip(p.vales(), p.keys()))[1]
AttributeError: 'dict' object has no attribute 'vales'

======================================================================
FAIL: test_max (__main__.TestsForSnakeStats)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-5-b021a2883284>", line 21, in test_max
    self.assertTrue(result1==result2)
AssertionError: False is not true

======================================================================
FAIL: test_mean (__main__.TestsForSnakeStats)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-5-b021a2883284>", line 8, in test_mean
    self.assertEqual(result1, result2)
AssertionError: 23.444444444444443 != 30.666666666666668

----------------------------------------------------------------------
Ran 3 tests in 0.005s

FAILED (failures=2, errors=1)
<unittest.main.TestProgram at 0x17193a7c4c8>
```

Set 1 passed Screenshot:



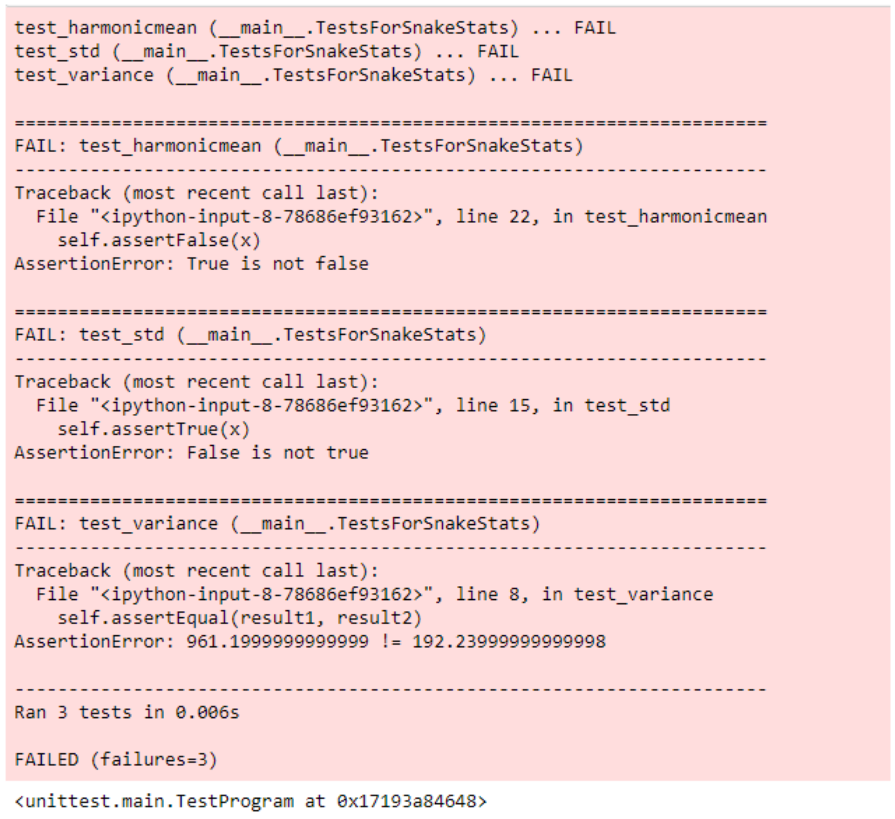
```
test_max (__main__.TestsForSnakeStats) ... ok
test_mean (__main__.TestsForSnakeStats) ... ok
test_mode (__main__.TestsForSnakeStats) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.003s

OK
<unittest.main.TestProgram at 0x17193a6f748>
```

Test 4(set-2): this test is to calculate the variance of our list, then compare it with statistic library output using assert-equal function, we are straight forwardly using variance formula $\frac{\sum_{i=0}^{n}(list[i]-mean)^2}{n}$ to calculate it. Here our first algorithm failed because we forgot to divide the final result with n.

Test 5(set-2): this test is to calculate the standard deviation of our list, then compare it with numpy library output using assert-True function, we are straight forwardly using standard deviation formula $\sqrt{\frac{\sum_{i=0}^{n}(list[i]-mean)^2}{n}}$ to calculate it. Here our first algorithm failed because we forgot to take the under root of our final result.

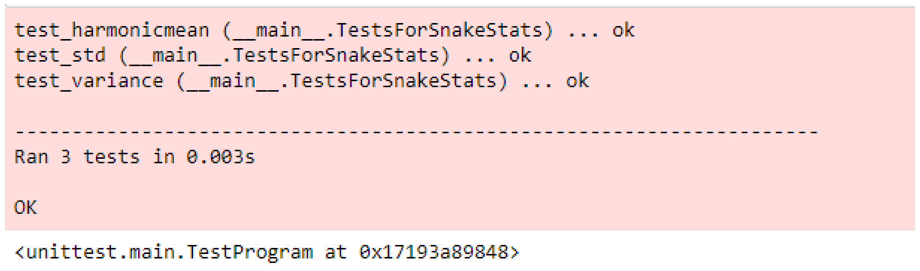
Test 6(set-2): this test is to calculate the harmonic mean of our list, then compare it with statistic library output using assert-False function, we are straight forwardly using harmonic mean formula $\frac{n}{\sum_{i=0}^{n} 1/list[i]}$ to calculate it. Here our first algorithm failed because we miscalculated the length of our list(using counter variable).

Set 2 Failed Screenshot:



```
test_harmonicmean (__main__.TestsForSnakeStats) ... FAIL
test_std (__main__.TestsForSnakeStats) ... FAIL
test_variance (__main__.TestsForSnakeStats) ... FAIL

======================================================================
FAIL: test_harmonicmean (__main__.TestsForSnakeStats)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-8-78686ef93162>", line 22, in test_harmonicmean
    self.assertFalse(x)
AssertionError: True is not false

======================================================================
FAIL: test_std (__main__.TestsForSnakeStats)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-8-78686ef93162>", line 15, in test_std
    self.assertTrue(x)
AssertionError: False is not true

======================================================================
FAIL: test_variance (__main__.TestsForSnakeStats)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-8-78686ef93162>", line 8, in test_variance
    self.assertEqual(result1, result2)
AssertionError: 961.1999999999999 != 192.23999999999998

----------------------------------------------------------------------
Ran 3 tests in 0.006s

FAILED (failures=3)
<unittest.main.TestProgram at 0x17193a84648>
```

Set 2 Passed Screenshot:



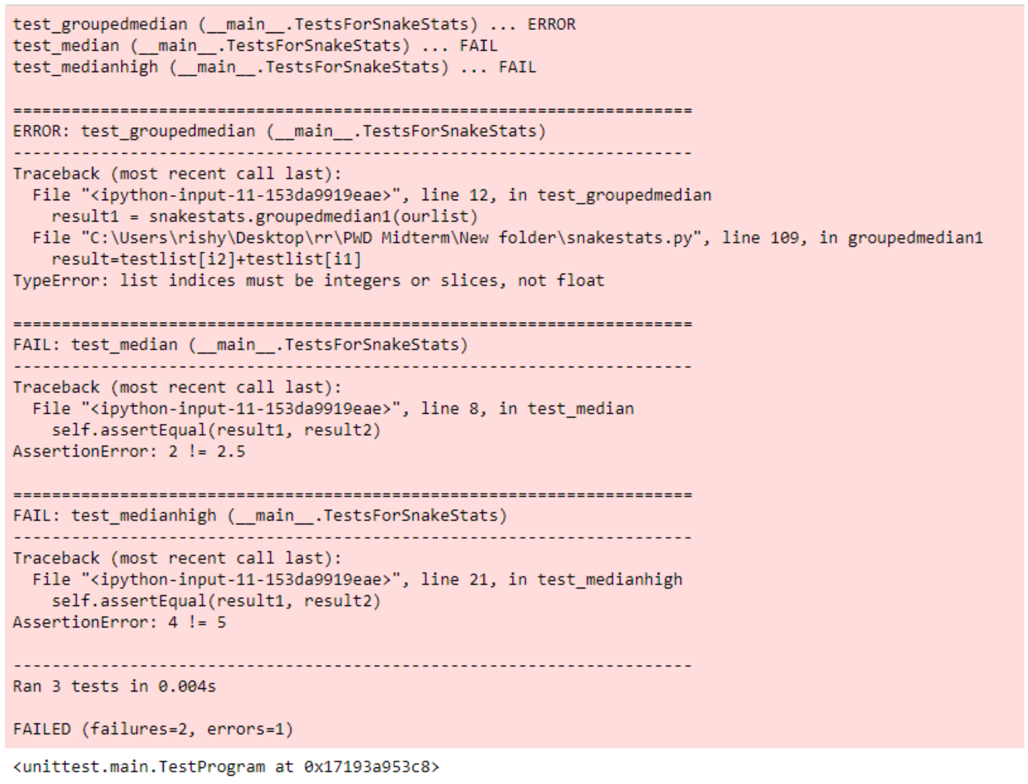
```
test_harmonicmean (__main__.TestsForSnakeStats) ... ok
test_std (__main__.TestsForSnakeStats) ... ok
test_variance (__main__.TestsForSnakeStats) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.003s

OK
<unittest.main.TestProgram at 0x17193a89848>
```

Test 7(set-3): this test is to find the median number of our list, then compare it with statistic library output using assert-Equal function, in the first algorithm our approach was to maintain 2 counter variables, one for very-first index and another for last, and keep on incrementing & decrementing them respectively until we reach the middle point, and in second algorithm we adapted our code to work even length too.

Test 8(set-3): this test is to calculate grouped-median of our list, then compare it with statistic library output using assert-False function, here our approach is to divide the list into two equal parts, then first we'll do the summation of last and first element respectively and then divide the result by two. Here our first algorithm failed because we forgot to convert float number to integer before passing to list index, Hence showing type-error.

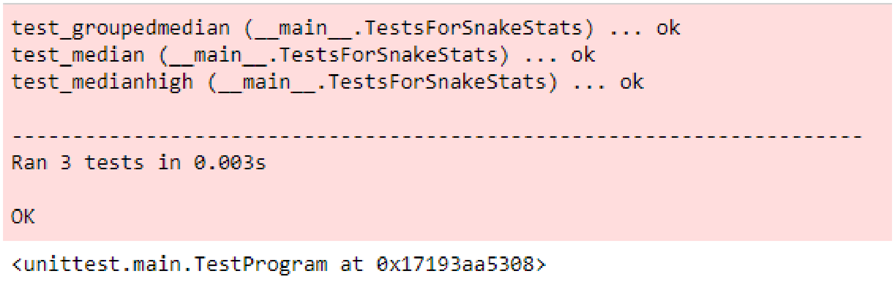
Test 9(set-3): this test is to find the higher-median of our list, then compare it with statistic library output using assert-Equal function, here we keep on deleting first and last element our list until the list size is 2 and then return the last element, our first algorithm failed because our while-loop counter wasn't able to alternate properly.

Set 3 Failed Screenshot:



```
test_groupedmedian (__main__.TestsForSnakeStats) ... ERROR
test_median (__main__.TestsForSnakeStats) ... FAIL
test_medianhigh (__main__.TestsForSnakeStats) ... FAIL

======================================================================
ERROR: test_groupedmedian (__main__.TestsForSnakeStats)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-11-153da9919eae>", line 12, in test_groupedmedian
    result1 = snakestats.groupedmedian1(ourlist)
  File "C:\Users\rishy\Desktop\rr\PWD Midterm\New folder\snakestats.py", line 109, in groupedmedian1
    result=testlist[i2]+testlist[i1]
TypeError: list indices must be integers or slices, not float

======================================================================
FAIL: test_median (__main__.TestsForSnakeStats)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-11-153da9919eae>", line 8, in test_median
    self.assertEqual(result1, result2)
AssertionError: 2 != 2.5

======================================================================
FAIL: test_medianhigh (__main__.TestsForSnakeStats)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-11-153da9919eae>", line 21, in test_medianhigh
    self.assertEqual(result1, result2)
AssertionError: 4 != 5

----------------------------------------------------------------------
Ran 3 tests in 0.004s

FAILED (failures=2, errors=1)
<unittest.main.TestProgram at 0x17193a953c8>
```

Set 3 Passed Screenshot:



```
test_groupedmedian (__main__.TestsForSnakeStats) ... ok
test_median (__main__.TestsForSnakeStats) ... ok
test_medianhigh (__main__.TestsForSnakeStats) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.003s

OK
<unittest.main.TestProgram at 0x17193aa5308>
```