Test-driven development with snake-stats

In this worksheet, you will carry out a test-driven development workflow in the Python language using the unittest module.

Set up your environment

We have set up a Jupyter notebooks environment for you to use to carry out the work in this worksheet. We recommend that you use this environment to do this lab.

You can also carry out the work on your own local machine if you have Python 3 installed. Instructions on how to set up a Python 3 environment on your own machine are beyond the scope of this worksheet. We recommend that you use anaconda if you want to set up a local Python development environment: https://www.anaconda.com/products/individual.

Start the lab environment application

It is simple to launch a lab exercise. You only need to click on the button "Start" below the activity title to enter a lab environment. Let's explore this lab activity. Go ahead and click on the "Start" button!

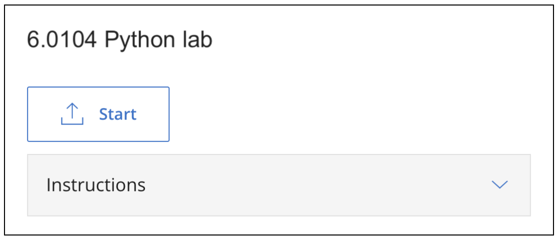


Figure 1: Coursera "Start" button

Create the snakestep module

In the file browser window of the jupyter notebook environment, create a new text file (not a Python file!):

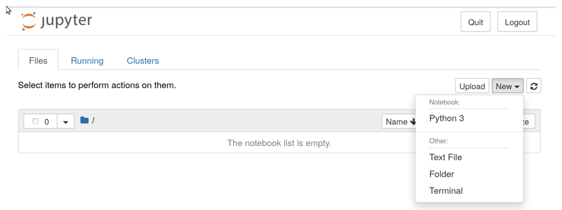


Figure 2: Jupyter new file

The file should open in a new tab. Rename it to snakestats.py by clicking on the filename at the top of the window. It is probably untitled.txt at the moment.

Now you have created the module.

Write the first failing test

The first step in test-driven development is to write a failing test.

Create another new file in the Jupyter notebook folder that you created snakestats.py in. This time, the file type should be Python, from the notebook section of the new file menu.

Put the following code in your new notebook in the first cell:

```
import unittest
import snakestats

class TestsForSnakeStats(unittest.TestCase):

    def test_mean(self):
        res = snakestats.mean([1,2,3])
        self.assertEqual(res, 2.0)

unittest.main(argv=['ignored', '-v'], exit=False)
```

Run it - you should see some output like this:

```
test_mean (__main__.TestsForSnakeStats) ... ERROR

======================================================================
ERROR: test_mean (__main__.TestsForSnakeStats)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-3-1eba45210d16>", line 7, in test_mean
    res = snakestats.mean([1,2,3])
AttributeError: module 'snakestats' has no attribute 'mean'

----------------------------------------------------------------------
Ran 1 test in 0.002s

FAILED (errors=1)
```

If you see an error like this:

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-1-1eba45210d16> in <module>
      1 import unittest
----> 2 import snakestats
      3 
      4 class TestsForSnakeStats(unittest.TestCase):
      5 

ModuleNotFoundError: No module named 'snakestats'
```

That means you do not have a file called snakestats.py in the same folder as the notebook file, so you need to correctr that.

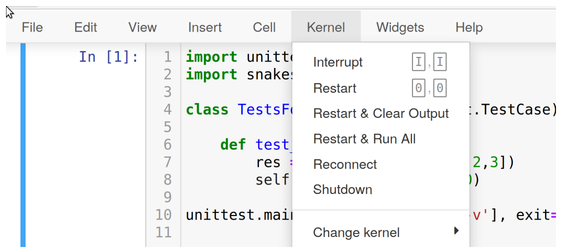
Write some code to pass the test

Next we are going to write some code to pass this test. Put this code into snakestats.py:

```
def mean(vals):
    sum = vals[0] + vals[1] + vals[2]
    return sum / 3
```

We know that you can use numpy etc. to calculate these values, and we know this is a naive implementation of mean, but we are doing it ourselves so that we can learn about test-driven development.

Now re-run your jupyter notebook code. To make sure the notebook reloads the snakestats module, call returt and run all from the kernel menu in Jupyter notebooks:



```
import unitt
import snake

class TestsF                        .TestCase)

    def test_
        res =                       2,3])
        self                        )

unittest.mai                        v'], exit=
```

Figure 3: Restart and run all menu

Write another test

Now let's write another failing test. We will keep the first test though. Update your test notebook cell so it looks like this:

```
import unittest
import snakestats

class TestsForSnakeStats(unittest.TestCase):

    def test_mean_3_values(self):
        res = snakestats.mean([1,2,3])
        self.assertEqual(res, 2.0)
    def test_mean_2_values(self):
        res = snakestats.mean([1,2])
        self.assertEqual(res, 1.5)

unittest.main(argv=['ignored', '-v'], exit=False)
```

Run the cell again (no need to restart the kernel as smakestats has not changed). You should see some output like this:

```
test_mean_2_values (__main__.TestsForSnakeStats) ... ERROR
test_mean_3_values (__main__.TestsForSnakeStats) ... ok

======================================================================
ERROR: test_mean_2_values (__main__.TestsForSnakeStats)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-1-ace42a524947>", line 10, in test_mean_2_values
    res = snakestats.mean([1,2])
  File "/home/matthew/src/bsc-cs/software-design-and-development/
  sdd-teaching-materials/src/unit-testing/labs/snakestats/src/
  temp/snakestats.py", line 10, in mean
    sum = vals[0] + vals[1] + vals[2]
IndexError: list index out of range

----------------------------------------------------------------------
Ran 2 tests in 0.002s

FAILED (errors=1)
```

Update the code so it passes the new test

Now replace the code in snakestats.py with this:

```
def mean(vals):
    sum = 0.0
    for i in range(len(vals)):
        sum += vals[i]

    return sum / len(vals)
```

Save snakestates.py and restart and run-all from the notebook menu on the unit test notebook.

You should now see some output like this:

```
test_mean_2_values (__main__.TestsForSnakeStats) ... ok
test_mean_3_values (__main__.TestsForSnakeStats) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK
```

Great - we are making progress now.

Challenge: write another failing test

Can you think of a test that the new mean code will fail? Write this test, then write code to make it pass.

Time for a new function - getMax

Now we are going to work on another statistical function - getMax. Start by writing a failing test:

```
def test_getMax(self):
    theMax = snakestats.getMax([6,5,4,3,2,1])
    self.assertEqual(theMax, 6)
```

Basic implementation of getMax

Now the minimal implementation to pass this test:

```
def getMax(vals):
    return vals[0]
```

./pagebreak

Another failing test for getMax

```
def test_getMaxLast(self):
    theMax = snakestats.getMax([6,5,4,3,2,7])
    self.assertEqual(theMax, 7)
```

Now update the getMax implementation to pass this test.

```
def getMax(vals):
    maxSeen = 0
    for i in range(len(vals)):
        if vals[i] > maxSeen:
            maxSeen = vals[i]
    return maxSeen
```

Challenge: think of a test that the getMax implementation will fail

Can you think of a test which the getMax implementation above will fail?

Update getMax to pass your new test.