Assignment 7 question 1 & 2

November 25, 2018

1 Question 1

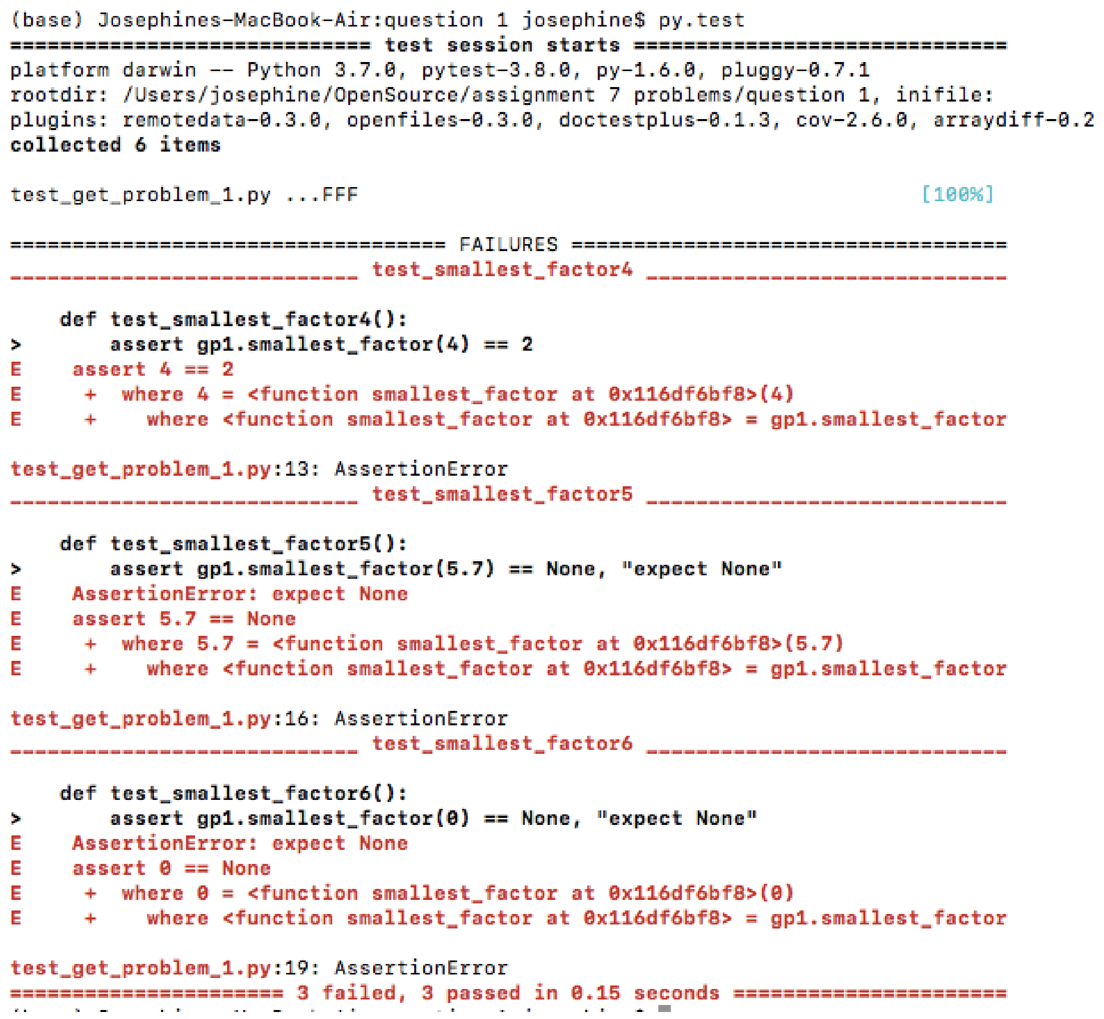
Problem 1: writing unit tests for the following function

```
In [10]: def smallest_factor(n):
             """Return the smallest prime factor of the positive integer n."""
             if n == 1: return 1
             for i in range(2, int(n**.5)):
                 if n % i == 0: return i
             return n
```

```
In [22]: #Unit tests for the function above:
         import get_problem_1 as gp1

         def test_smallest_factor1():
             assert gp1.smallest_factor(1) == 1

         def test_smallest_factor2():
             assert gp1.smallest_factor(2) == 2

         def test_smallest_factor3():
             assert gp1.smallest_factor(3) == 3

         def test_smallest_factor4():
             assert gp1.smallest_factor(4) == 2

         def test_smallest_factor5():
             assert gp1.smallest_factor(5.7) == None, "expect None"

         def test_smallest_factor6():
             assert gp1.smallest_factor(0) == None, "expect None"
```

```
In [12]: #Running the unit tests results in the following:
         from IPython.display import Image
         Image(filename='results_test1.png')
```

```
Out[12]:
```

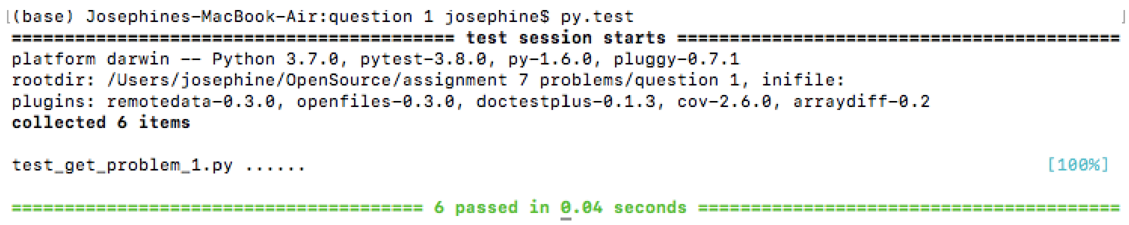


```
(base) Josephines-MacBook-Air:question 1 josephine$ py.test
============================= test session starts ==============================
platform darwin -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: /Users/josephine/OpenSource/assignment 7 problems/question 1, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 6 items

test_get_problem_1.py ...FFF                                             [100%]

=================================== FAILURES ===================================
_____________________________ test_smallest_factor4 _____________________________

    def test_smallest_factor4():
>       assert gp1.smallest_factor(4) == 2
E    assert 4 == 2
E     +  where 4 = <function smallest_factor at 0x116df6bf8>(4)
E     +    where <function smallest_factor at 0x116df6bf8> = gp1.smallest_factor

test_get_problem_1.py:13: AssertionError
_____________________________ test_smallest_factor5 _____________________________

    def test_smallest_factor5():
>       assert gp1.smallest_factor(5.7) == None, "expect None"
E    AssertionError: expect None
E    assert 5.7 == None
E     +  where 5.7 = <function smallest_factor at 0x116df6bf8>(5.7)
E     +    where <function smallest_factor at 0x116df6bf8> = gp1.smallest_factor

test_get_problem_1.py:16: AssertionError
_____________________________ test_smallest_factor6 _____________________________

    def test_smallest_factor6():
>       assert gp1.smallest_factor(0) == None, "expect None"
E    AssertionError: expect None
E    assert 0 == None
E     +  where 0 = <function smallest_factor at 0x116df6bf8>(0)
E     +    where <function smallest_factor at 0x116df6bf8> = gp1.smallest_factor

test_get_problem_1.py:19: AssertionError
====================== 3 failed, 3 passed in 0.15 seconds ======================
```

```
In [13]: # As seen above, the function fails in three tests: when n = 4, when n = 5.7, and whe
         # To correct this, we can change the function to the following:
         def smallest_factor(n):
             """Return the smallest prime factor of the positive integer n."""
             if n <= 0 or type(n)!= int : return None
             if n == 1: return 1
             for i in range(2, int(n**.5)+1):
                     if n % i == 0: return i
             return n
```

```
In [14]: #Running the unit tests with the new function results in the following:
         from IPython.display import Image
         Image(filename='results_test1b.png')
```

Out[14]:

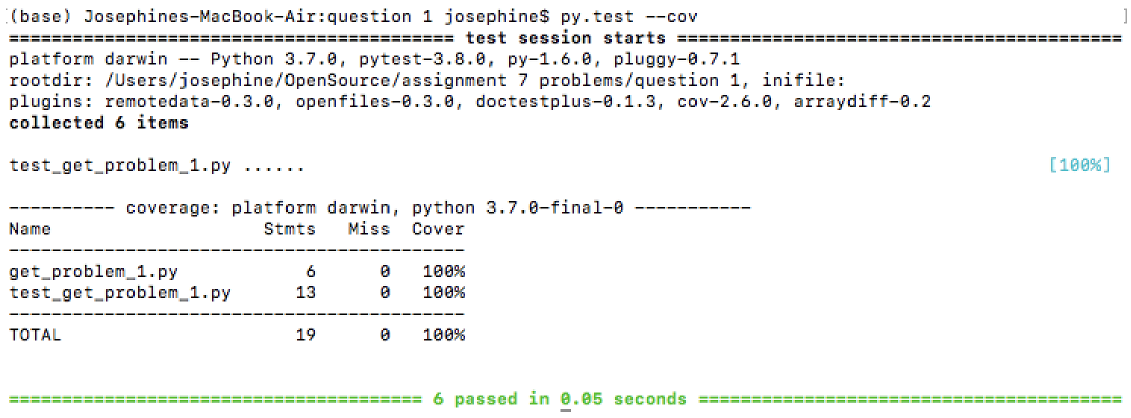


```
(base) Josephines-MacBook-Air:question 1 josephine$ py.test
========================================= test session starts ==========================================
platform darwin -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: /Users/josephine/OpenSource/assignment 7 problems/question 1, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 6 items

test_get_problem_1.py ......                                                                     [100%]

======================================= 6 passed in 0.04 seconds =======================================
```

Problem 2: checking coverage and writing unit tests

```
In [16]: #Checking coverage of the function in question 1, results in the following:
         from IPython.display import Image
         Image(filename='results_test1c.png')
```

```
Out[16]:
```



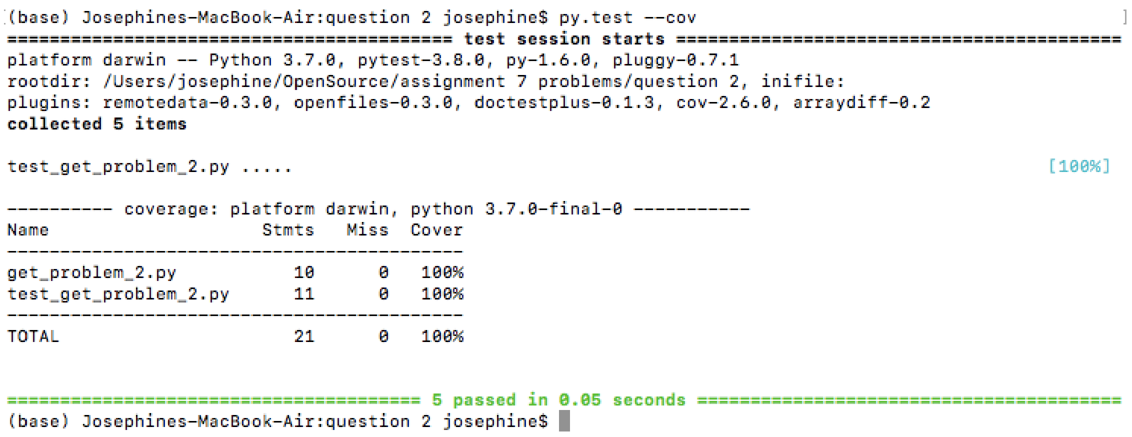
```
(base) Josephines-MacBook-Air:question 1 josephine$ py.test --cov
========================================= test session starts ==========================================
platform darwin -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: /Users/josephine/OpenSource/assignment 7 problems/question 1, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 6 items

test_get_problem_1.py ......                                                                     [100%]

---------- coverage: platform darwin, python 3.7.0-final-0 -----------
Name                     Stmts   Miss  Cover
--------------------------------------------
get_problem_1.py             6      0   100%
test_get_problem_1.py       13      0   100%
--------------------------------------------
TOTAL                       19      0   100%


======================================= 6 passed in 0.05 seconds =======================================
```

```
In [ ]: #Writing unit tests for the following function:
        def month_length(month, leap_year=False):
            """Return the number of days in the given month."""
            if month in {"September", "April", "June", "November"}:
                return 30
            elif month in {"January", "March", "May", "July", "August", "October", "December"}
                return 31
            if month == "February":
                if not leap_year:
                    return 28
            else:
                    return 29
            else:
                    return None
```

```
In [ ]: #Unit tests for the function above:
        import get_problem_2 as gp2

        def test_month_length1():
            assert gp2.month_length("September") == 30, "expect 30"

        def test_month_length2():
            assert gp2.month_length("January") == 31, "expect 31"

        def test_month_length3():
            assert gp2.month_length("February") == 28, "expect 28"

        def test_month_length4():
            assert gp2.month_length("February", leap_year=True) == 29, "expect 29"

        def test_month_length5():
            assert gp2.month_length(1) == None
```

```
In [17]: #Running the unit tests and checking coverage results in the following:
         from IPython.display import Image
         Image(filename='results_test2.png')
```

Out[17]:



```
(base) Josephines-MacBook-Air:question 2 josephine$ py.test --cov
=========================================== test session starts ============================================
platform darwin -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: /Users/josephine/OpenSource/assignment 7 problems/question 2, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 5 items

test_get_problem_2.py .....                                                                          [100%]

---------- coverage: platform darwin, python 3.7.0-final-0 -----------
Name                    Stmts   Miss  Cover
-------------------------------------------
get_problem_2.py           10      0   100%
test_get_problem_2.py      11      0   100%
-------------------------------------------
TOTAL                      21      0   100%


======================================= 5 passed in 0.05 seconds ========================================
(base) Josephines-MacBook-Air:question 2 josephine$
```

Problem 3: writing comprehensive test for the following function

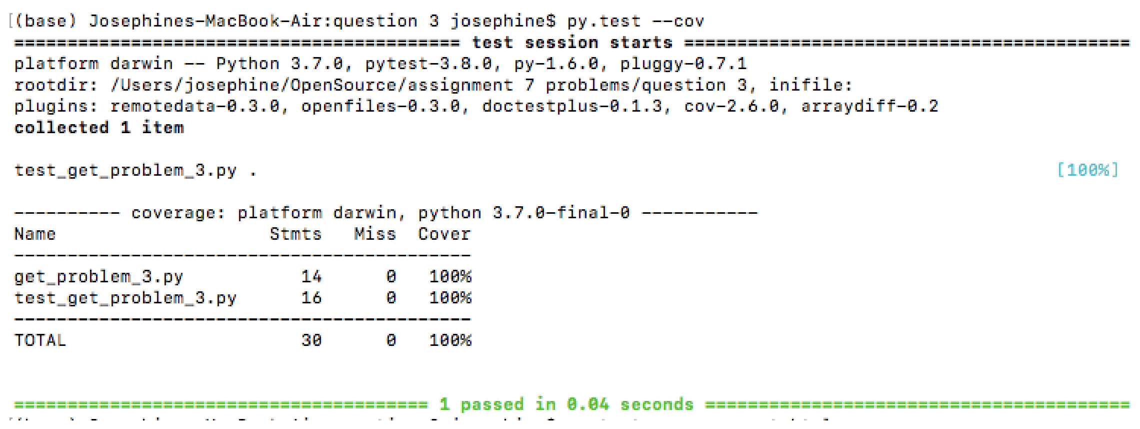
```
In [ ]: def operate(a, b, oper):
        """Apply an arithmetic operation to a and b."""
            if type(oper) is not str:
                raise TypeError("oper must be a string")
            elif oper == '+':
                return a + b
            elif oper == '-':
```

```
            return a - b
        elif oper == '*':
            return a * b
        elif oper == '/':
            if b == 0:
                raise ZeroDivisionError("division by zero is undefined")
            return a / b
        raise ValueError("oper must be one of '+', '/', '-', or '*'")
```

```
In [ ]: #Unit tests for the above function
        import pytest
        import get_problem_3 as gp3

        def test_operate():
            assert gp3.operate(2, 5, '+') == 7
            assert gp3.operate(5, 5, '-') == 0
            assert gp3.operate(2, 3, '*') == 6
            assert gp3.operate(6, 2, '/') == 3
            with pytest.raises(ZeroDivisionError) as excinfo:
                gp3.operate(6, 0, '/')
            assert excinfo.value.args[0] == "division by zero is undefined"
            with pytest.raises(ValueError) as excinfo:
                gp3.operate(2, 3, 'g')
            assert excinfo.value.args[0] == "oper must be one of '+', '/', '-', or '*'"
            with pytest.raises(TypeError) as excinfo:
                gp3.operate(2, 3, 4)
            assert excinfo.value.args[0] == "oper must be a string"
```

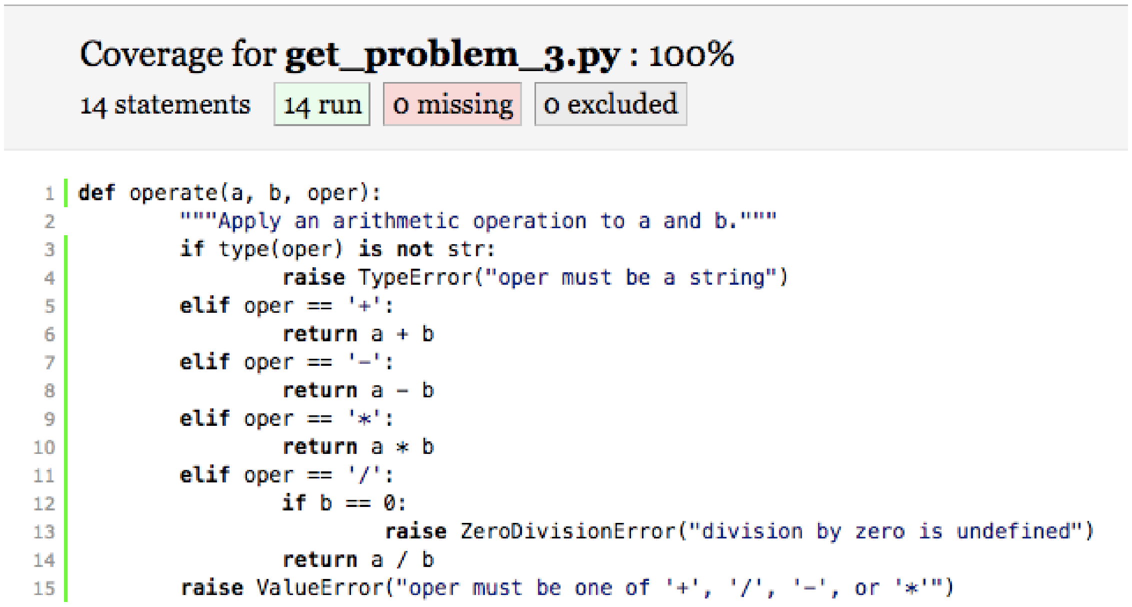
```
In [18]: #Running the unit tests and checking coverage results in the following:
         from IPython.display import Image
         Image(filename='results_test3.png')
```

Out[18]:



```
[(base) Josephines-MacBook-Air:question 3 josephine$ py.test --cov
========================================== test session starts ==========================================
platform darwin -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: /Users/josephine/OpenSource/assignment 7 problems/question 3, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 1 item

test_get_problem_3.py .                                                                          [100%]

---------- coverage: platform darwin, python 3.7.0-final-0 -----------
Name                       Stmts   Miss  Cover
----------------------------------------------
get_problem_3.py              14      0   100%
test_get_problem_3.py         16      0   100%
----------------------------------------------
TOTAL                         30      0   100%


======================================= 1 passed in 0.04 seconds ========================================
In [19]: #Obtaining html documents to check coverage
         from IPython.display import Image
         Image(filename='results_test3a.png')
```

```
Out[19]:
```



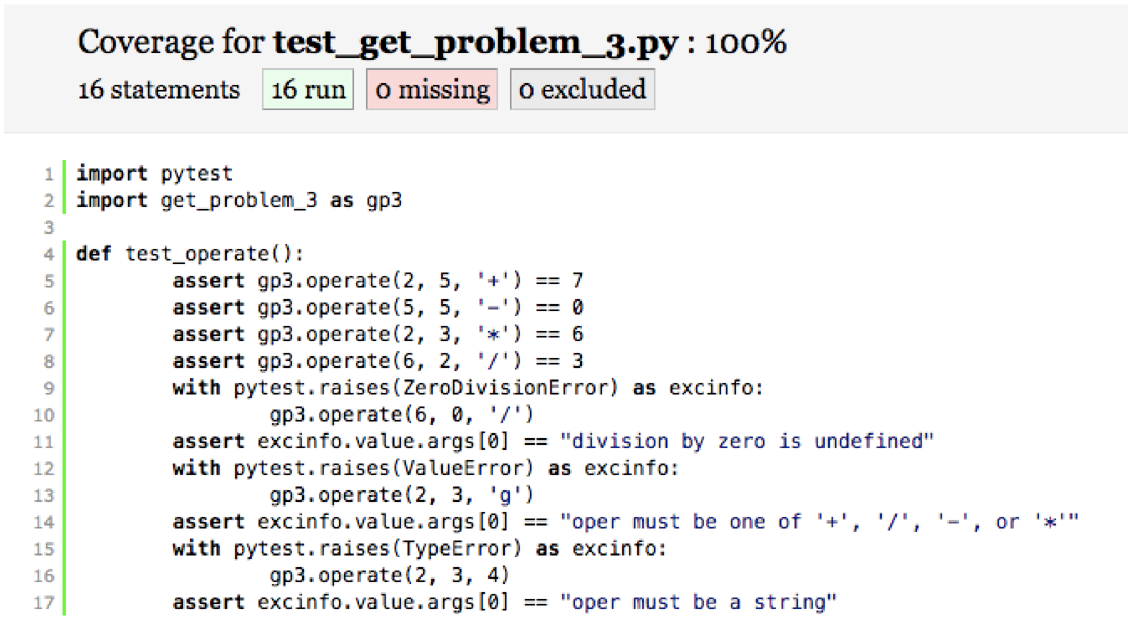
Coverage for **get_problem_3.py** : 100%

14 statements | 14 run | 0 missing | 0 excluded

```
def operate(a, b, oper):
        """Apply an arithmetic operation to a and b."""
        if type(oper) is not str:
                raise TypeError("oper must be a string")
        elif oper == '+':
                return a + b
        elif oper == '-':
                return a - b
        elif oper == '*':
                return a * b
        elif oper == '/':
                if b == 0:
                        raise ZeroDivisionError("division by zero is undefined")
                return a / b
        raise ValueError("oper must be one of '+', '/', '-', or '*'")
```

```
In [20]: from IPython.display import Image
         Image(filename='results_test3b.png')
```

```
Out[20]:
```



Coverage for **test_get_problem_3.py** : 100%

16 statements | 16 run | 0 missing | 0 excluded

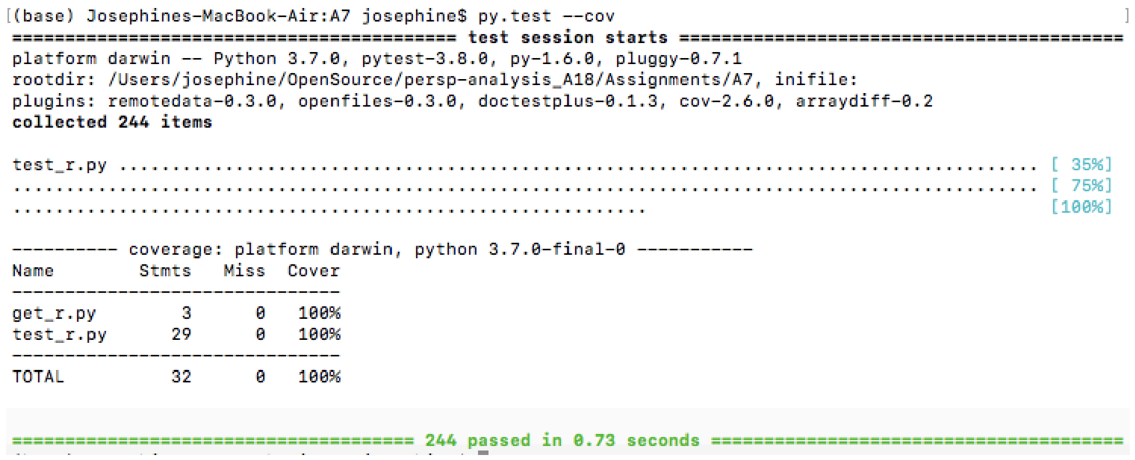
```
import pytest
import get_problem_3 as gp3

def test_operate():
        assert gp3.operate(2, 5, '+') == 7
        assert gp3.operate(5, 5, '-') == 0
        assert gp3.operate(2, 3, '*') == 6
        assert gp3.operate(6, 2, '/') == 3
        with pytest.raises(ZeroDivisionError) as excinfo:
                gp3.operate(6, 0, '/')
        assert excinfo.value.args[0] == "division by zero is undefined"
        with pytest.raises(ValueError) as excinfo:
                gp3.operate(2, 3, 'g')
        assert excinfo.value.args[0] == "oper must be one of '+', '/', '-', or '*'"
        with pytest.raises(TypeError) as excinfo:
                gp3.operate(2, 3, 4)
        assert excinfo.value.args[0] == "oper must be a string"
```

2 Question 2

```
In [ ]: #Function for theory of firms
        def get_r(K, L, alpha, Z, delta):
        ''''This function generates the interest rate or vector of interest rates'''
            r = (alpha * Z * (L / K) ** (1 - alpha)) - delta
            return r
```

```
In [21]: #Checking coverage of the function
         from IPython.display import Image
         Image(filename='results_question2.png')
```

```
Out[21]:
```



```
[(base) Josephines-MacBook-Air:A7 josephine$ py.test --cov                                                                  ]
========================================== test session starts ==========================================
platform darwin -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: /Users/josephine/OpenSource/persp-analysis_A18/Assignments/A7, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 244 items

test_r.py ...................................................................................... [ 35%]
................................................................................................ [ 75%]
............................................................                                     [100%]

---------- coverage: platform darwin, python 3.7.0-final-0 -----------
Name         Stmts   Miss  Cover
--------------------------------
get_r.py         3      0   100%
test_r.py       29      0   100%
--------------------------------
TOTAL           32      0   100%


===================================== 244 passed in 0.73 seconds =====================================
```

The function passes with 100% coverage results.