Answers(Q1 and Q2)_Assignment 7

November 25, 2018

1 Assignment 7

MACS 30000, Dr. Evans

Sixue Liu Due Monday, Nov. 26 at 11:30 AM

1. Unit Testing in Python (3 points).

Problem 1 This is the original function:

```
In [7]: 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
```

This is the unit test I wrote for this function:

```
In [ ]: import pers

        def test_pers_prime():
            assert pers.smallest_factor(7) == 7, "failed on prime number"

        def test_pers_even():
            assert pers.smallest_factor(8) == 2, "failed on even number"

        def test_pers_odd():
            assert pers.smallest_factor(9) == 3, "failed on odd number"
```

This is the result I got after running py.test(Q1 P1):

Footnote: The image may not match the original text when displayed in PDF, please follow the picture title when you feel confused.

```
In [ ]: #This is the function after correcting:
        def smallest_factor(n):
        """Return the smallest prime factor of the positive integer n."""
```

```
(base) C:\Users\Graci\Desktop\problem 1>py.test
============================= test session starts =============================
platform win32 -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: C:\Users\Graci\Desktop\problem 1, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, arraydiff-0.2
collected 3 items

test_p1.py .FF                                                           [100%]

================================== FAILURES ===================================
_____________________________ test_pers_even ______________________________

    def test_pers_even():
>       assert p1.smallest_factor(8) == 2, "failed on even number"
E       AssertionError: failed on even number
E       assert 8 == 2
E        +  where 8 = <function smallest_factor at 0x000001E6231112F0>(8)
E        +    where <function smallest_factor at 0x000001E6231112F0> = p1.smallest_factor

test_p1.py:13: AssertionError
_____________________________ test_pers_odd _______________________________

    def test_pers_odd():
>       assert p1.smallest_factor(9) == 3, "failed on odd number"
E       AssertionError: failed on odd number
E       assert 9 == 3
E        +  where 9 = <function smallest_factor at 0x000001E6231112F0>(9)
E        +    where <function smallest_factor at 0x000001E6231112F0> = p1.smallest_factor

test_p1.py:16: AssertionError
==================== 2 failed, 1 passed in 0.19 seconds =====================
```

Q1 P1

```
(base) C:\Users\Graci\Desktop\problem 1>py.test
============================= test session starts =============================
platform win32 -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: C:\Users\Graci\Desktop\problem 1, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, arraydiff-0.2
collected 3 items

test_p1.py ...                                                           [100%]

========================== 3 passed in 0.05 seconds ===========================
```

Q1 P2

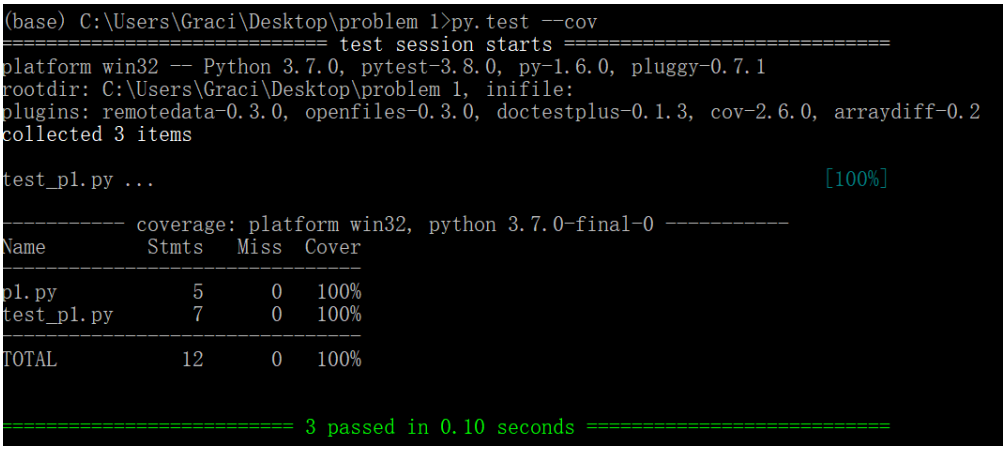
```
    if n == 1: return 1
    for i in range(2, int(n**.5) + 1):
        if n % i == 0: return i
    return n
```

This is the new result I got after running the py.test(Q1 P2):

Answer description: The reason why the original function cannot pass the test is 'range' in Python only includes the lower bound. In this case, 'i' needs to iterate from the lower bound to the upper bound, inclusively. The original function doesn't include the upper bound. Therefore, the way to fix this is to add 1 to the upper bound, now 'i' could also iterate the upper bound.

Problem 2 This is the result coverage of problem 1(Q1 P3):

```
In [1]: #This is the function of problem 2
        def month_length(month, leap_year=False):
            """Return the number of days in the given month."""
```



Q1

```
    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 [ ]: #This is the test function I wrote for problem 2
        import p2

        def test_p2_month():
            assert p2.month_length('January') == 31, "failed on January"
            assert p2.month_length('April') == 30, "failed on April"
            assert p2.month_length('February', leap_year=True) == 29, "failed on February_leap
            assert p2.month_length('February', leap_year=False) == 28, "failed on February_not
            assert p2.month_length('Other') == None, "failed on other_cases"
```

This is the result after running py.test and py.test --cov(Q1 P4):

Answer Description: The code in problem 2 has several different cases. Generally, we need to consider the months with 30 days and others with 31 days. Also, February is a special case. The length of February depends on whether it is the leap year or not. Therefore, I need to design the test code for both cases. Also, if the input is other word, we need to test if the result is None.

Problem 3

```
In [ ]: #This is the function from problem 3
        def operate(a, b, oper):
```

```
(base) C:\Users\Graci\Desktop\problem 2>py.test
============================= test session starts =============================
platform win32 -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: C:\Users\Graci\Desktop\problem 2, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 1 item

p2_test.py .                                                             [100%]

========================== 1 passed in 0.05 seconds ===========================

(base) C:\Users\Graci\Desktop\problem 2>py.test --cov
============================= test session starts =============================
platform win32 -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: C:\Users\Graci\Desktop\problem 2, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 1 item

p2_test.py .                                                             [100%]

---------- coverage: platform win32, python 3.7.0-final-0 -----------
Name         Stmts   Miss  Cover
--------------------------------
p2.py           10      0   100%
p2_test.py       7      0   100%
--------------------------------
TOTAL           17      0   100%


========================== 1 passed in 0.06 seconds ===========================
```

Q1 P4

```
    """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 [ ]: #This is the unit test I wrote for problem 3
        import pytest
        import p3

        def test_p3():
            assert p3.operate(4, 2, '+') == 6, "failed on addition"
            assert p3.operate(2, 4, '-') == -2, "failed on subtraction"
            assert p3.operate(4, 2, '*') == 8, "failed on multiplication"
            assert p3.operate(9, 3, '/') == 3, "failed on division"
            with pytest.raises(TypeError) as excinfo1:
                p3.operate(4, 0, 8)
```

```
(base) C:\Users\Graci\Desktop\problem 3>py.test
============================= test session starts =============================
platform win32 -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: C:\Users\Graci\Desktop\problem 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

p3_test.py .                                                             [100%]

========================== 1 passed in 0.05 seconds ===========================
```

Q1 P5

```
(base) C:\Users\Graci\Desktop\problem 3>py.test --cov
============================= test session starts =============================
platform win32 -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: C:\Users\Graci\Desktop\problem 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

p3_test.py .                                                             [100%]

---------- coverage: platform win32, python 3.7.0-final-0 ----------
Name         Stmts   Miss  Cover
--------------------------------
p3.py           14      0   100%
p3_test.py      16      0   100%
--------------------------------
TOTAL           30      0   100%


========================== 1 passed in 0.05 seconds ===========================
```

Q1 P6

```
    assert excinfo1.value.args[0] == "oper must be a string"
    with pytest.raises(ZeroDivisionError) as excinfo2:
        p3.operate(4, 0, '/')
    assert excinfo2.value.args[0] == "division by zero is undefined"
    with pytest.raises(ValueError) as excinfo3:
        p3.operate(4, 0, '&')
    assert excinfo3.value.args[0] == "oper must be one of '+', '/', '-', or '*'"
```

This is the result after running py.test(Q1 P5):

This is the result after running py.test --cov(Q1 P6):

This is the result using cov-report tool to confirm that I have full coverage of this function(Q1 P7 and Q1 P8):

Answer Description: In the operate function, it raises three different errors: TypeError, ZeroDivisionError and ValueError. Therefore, not only do we need to consider four common situations like addition, subtraction, multiplication and division, but we should include the situations raising these three errors. In raising three errors, the error message should be the same with the original function.

2. Test driven development (3 points).

```
In [2]: #This is the Python module:
        import numpy as np
```

Coverage for **p3.py** : 100%

14 statements | 14 run | 0 missing | 0 excluded

```
#!/usr/bin/env python
# coding: utf-8

# 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 '*'")

```

Q1 P7

Coverage for **p3_test.py** : 100%

16 statements | 16 run | 0 missing | 0 excluded

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:

import pytest
import p3

def test_p3():
    assert p3.operate(4, 2, '+') == 6, "failed on addition"
    assert p3.operate(2, 4, '-') == -2, "failed on subtraction"
    assert p3.operate(4, 2, '*') == 8, "failed on multiplication"
    assert p3.operate(9, 3, '/') == 3, "failed on division"
    with pytest.raises(TypeError) as excinfo1:
        p3.operate(4, 0, 8)
    assert excinfo1.value.args[0] == "oper must be a string"
    with pytest.raises(ZeroDivisionError) as excinfo2:
        p3.operate(4, 0, '/')
    assert excinfo2.value.args[0] == "division by zero is undefined"
    with pytest.raises(ValueError) as excinfo3:
        p3.operate(4, 0, '&')
    assert excinfo3.value.args[0] == "oper must be one of '+', '/', '-', or '*'"


```

Q1 P8

```
(base) C:\Users\Graci\Desktop\files\perspective\persp-analysis_A18\Assignments\A7>py.test
============================= test session starts =============================
platform win32 -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: C:\Users\Graci\Desktop\files\perspective\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_get_r.py ..............................................................  [ 23%]
........................................................................       [ 53%]
........................................................................       [ 82%]
.............................................                                  [100%]

======================== 244 passed in 0.54 seconds ========================
```

Q2 P9

```
(base) C:\Users\Graci\Desktop\files\perspective\persp-analysis_A18\Assignments\A7>py.test --cov
============================= test session starts =============================
platform win32 -- Python 3.7.0, pytest-3.8.0, py-1.6.0, pluggy-0.7.1
rootdir: C:\Users\Graci\Desktop\files\perspective\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_get_r.py ..............................................................  [ 23%]
........................................................................       [ 53%]
........................................................................       [ 82%]
.............................................                                  [100%]

---------- coverage: platform win32, python 3.7.0-final-0 ----------
Name            Stmts   Miss  Cover
-----------------------------------
get_r.py           11      0   100%
test_get_r.py      29      0   100%
-----------------------------------
TOTAL              40      0   100%


======================== 244 passed in 0.77 seconds ========================
```

Q2 P10

```python
def get_r(K, L, alpha, Z, delta):
    '''
    This function generates the interest rate or vector of interest rates
    '''
    assert alpha > 0 and alpha < 1,"failed on the range of capital share"
    assert delta >= 0 and delta < 1,"failed on the range of depreciation rate"
    assert Z > 0,"failed on the range of total factor productivity"

    r = alpha * Z * ( (L / K) ** (1 - alpha) ) - delta

    if (type(K) == float) and (type(L)== float):
        assert type(r)==float, "failed on 'if K and L are both scalars,this function sh

    if not np.isscalar(K) and not np.isscalar(L):
        assert not np.isscalar(r), "failed on 'if K and L are both vectors, this functi

    return r
```

This is the result after running py.test(Q2 P9):
This is the result after running py.test --cov(Q2 P10):
As we can see, the function I wrote passes all the tests and get fully covered.