

q1 and q2

November 26, 2018

1. Unit Testing in Python

Problem 1

```
In [11]: #original function
def smallest_factor(n):
    if n == 1: return 1
    for i in range(2, int(n**.5)):
        if n % i == 0: return i
    return n

In [ ]: #my test case
import smallest_factor

def test_sf():
    assert smallest_factor.smallest_factor(1) == 1, "failed on 1"
    assert smallest_factor.smallest_factor(4) == 2, "failed on square number"
    assert smallest_factor.smallest_factor(15) == 3, "failed on non-square number"
    assert smallest_factor.smallest_factor(7) == 7, "failed on prime number"
```

Below is the pytest result:

```
In [ ]: ===== test session starts =====
platform darwin -- Python 3.6.5, pytest-3.5.1, py-1.5.3, pluggy-0.6.0
rootdir: /Users/tianxinzheng/Desktop/MACS30000/persp-analysis_A18/Assignments, inifile
plugins: remotedata-0.2.1, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.1
collected 247 items

test_month_length.py . [ 0%]
test_oper.py . [ 0%]
test_r.py ..... [ 25%]
..... [ 55%]
..... [ 84%]
..... [ 99%]
test_sf.py F [100%]

===== FAILURES =====
----- test_sf -----
```

```

def test_sf():
    assert smallest_factor.smallest_factor(1) == 1, "failed on 1"
> assert smallest_factor.smallest_factor(4) == 2, "failed on square number"
E AssertionError: failed on square number
E assert 4 == 2
E + where 4 = <function smallest_factor at 0x1045938c8>(4)
E + where <function smallest_factor at 0x1045938c8> = smallest_factor.smallest_factor

test_sf.py:5: AssertionError
===== 1 failed, 246 passed in 0.51 seconds =====

```

I noticed that it failed on 4. This is because 2 is not in the range of `int(4 ** 0.5)`. Thus I modified the original function as follows:

```

In [10]: #modified function
def smallest_factor(n):
    if n == 1: return 1
    for i in range(2, int(n**.5) + 1):
        if n % i == 0: return i
    return n

```

Now it passed all the test.

```

In [ ]: ===== test session starts =====
platform darwin -- Python 3.6.5, pytest-3.5.1, py-1.5.3, pluggy-0.6.0
rootdir: /Users/tianxinzheng/Desktop/MACS30000/persp-analysis_A18/Assignments, inifile:
plugins: remotedata-0.2.1, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.1.0
collected 247 items

test_month_length.py .
test_oper.py .
test_r.py .....
.....
test_sf.py .

===== 247 passed in 0.46 seconds =====

```

Problem 2

```

In [ ]: ===== test session starts =====
platform darwin -- Python 3.6.5, pytest-3.5.1, py-1.5.3, pluggy-0.6.0
rootdir: /Users/tianxinzheng/Desktop/MACS30000/persp-analysis_A18/Assignments, inifile:
plugins: remotedata-0.2.1, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.1.0
collected 247 items

test_month_length.py .
test_oper.py .

```

```

test_r.py .....
.....
test_sf.py .

----- coverage: platform darwin, python 3.6.5-final-0 -----
Name                               Stmts  Miss  Cover
-----
get_r.py                           24     11    54%
month_length.py                    10      0   100%
oper.py                            14      0   100%
smallest_factor.py                 5      0   100%
test_month_length.py               7      0   100%
test_oper.py                       16      0   100%
test_r.py                          29      0   100%
test_sf.py                          6      0   100%
-----
TOTAL                             111     11    90%

```

My smallest_factor function already got full coverage.
This is the month_length function:

```

In [4]: def month_length(month, leap_year=False):
        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

```

This is my test_month_length_function:

```

In [ ]: import month_length

def test_month_length():
    assert month_length.month_length("January") == 31, \
        "failed on month with 31 days"
    assert month_length.month_length("September") == 30, \
        "failed on month with 30 days"
    assert month_length.month_length("February") == 28, \
        "failed on February with none-leap year"
    assert month_length.month_length("February", True) == 29, \
        "failed on February with leap year"

```

```

    assert month_length.month_length("not a month") == None, \
        "failed on month not given"

```

This is my test result:

```

In [ ]: ===== test session starts =====
platform darwin -- Python 3.6.5, pytest-3.5.1, py-1.5.3, pluggy-0.6.0
rootdir: /Users/tianxinzheng/Desktop/MACS30000/persp-analysis_A18/Assignments, inifile
plugins: remotedata-0.2.1, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.1
collected 247 items

test_month_length.py .
test_oper.py .
test_r.py .....
.....
test_sf.py .

===== 247 passed in 0.46 seconds =====

```

Problem 3

This is the operate function:

```

In [6]: def operate(a, b, oper):
        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 '*'")

```

This is my test_oper function:

```

In [ ]: import oper
        import pytest

        def test_oper():
            assert oper.operate(1, 3, "+") == 4, "failed on addition"
            assert oper.operate(1, 3, "-") == -2, "failed on subtraction"
            assert oper.operate(1, 3, "*") == 3, "failed on multiplication"
            assert oper.operate(1, 3, "/") == 1/3, "failed on division"
            with pytest.raises(TypeError) as excinfo1:

```

```

        oper.operate(1, 3, 4)
    assert excinfo1.value.args[0] == "oper must be a string"
    with pytest.raises(ZeroDivisionError) as excinfo2:
        oper.operate(1, 0, "/")
    assert excinfo2.value.args[0] == "division by zero is undefined"
    with pytest.raises(ValueError) as excinfo3:
        oper.operate(1, 3, ">")
    assert excinfo3.value.args[0] == "oper must be one of '+', '/', '-', or '*'"

```

I got full coverage of oper.py.

```

In [ ]: ===== test session starts =====
platform darwin -- Python 3.6.5, pytest-3.5.1, py-1.5.3, pluggy-0.6.0
rootdir: /Users/tianxinzheng/Desktop/MACS30000/persp-analysis_A18/Assignments, inifile:
plugins: remotedata-0.2.1, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2.0
collected 247 items

test_month_length.py .
test_oper.py .
test_r.py .....
.....
test_sf.py .

----- coverage: platform darwin, python 3.6.5-final-0 -----
Name                                Stmts   Miss  Cover
-----
get_r.py                             24      11    54%
month_length.py                       10       0   100%
oper.py                              14       0   100%
smallest_factor.py                     5       0   100%
test_month_length.py                   7       0   100%
test_oper.py                          16       0   100%
test_r.py                             29       0   100%
test_sf.py                             6       0   100%
-----
TOTAL                                111     11    90%

```

The coverage report is attached in the folder.

2 Test Driven Development

This is my get_r function:

```

In [9]: import numpy as np

def get_r(K, L, alpha, Z, delta):
    if type(K) is float and K <= 0:
        raise ValueError("K must be positive")
    if type(K) is np.array:
        if sum(K <= 0) > 0 :

```

```

        raise ValueError("K must be positive")
    if type(L) is float and L <= 0:
        raise ValueError("L must be positive")
    if type(L) is np.array:
        if sum(L <= 0) > 0 :
            raise ValueError("L must be positive")
    if Z <= 0:
        raise ValueError("Z must be positive")
    if alpha < 0 or alpha > 1:
        raise ValueError("alpha must be within 0 and 1")
    if delta < 0 or delta > 1:
        raise ValueError("delta must be within 0 and 1")

    r = alpha * Z * ((L / K) ** (1 - alpha)) - delta
    if type(K) is float and type(L) is float and type(r) is not float:
        raise TypeError("If K and L are scalars, r should also be a scaler")
    if type(K) is np.ndarray and type(L) is np.ndarray and type(r) is not np.ndarray:
        raise TypeError("If K and L are scalars, r should also be a scaler")

    return r

```

This is the pytest result:

```

In [ ]: ===== test session starts =====
platform darwin -- Python 3.6.5, pytest-3.5.1, py-1.5.3, pluggy-0.6.0
rootdir: /Users/tianxinzheng/Desktop/MACS30000/persp-analysis_A18/Assignments, inifile
plugins: remotedata-0.2.1, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 247 items

test_month_length.py .
test_oper.py .
test_r.py .....
.....
test_sf.py .

===== 247 passed in 0.46 seconds =====

```

This is the coverage result:

```

In [ ]: ===== test session starts =====
platform darwin -- Python 3.6.5, pytest-3.5.1, py-1.5.3, pluggy-0.6.0
rootdir: /Users/tianxinzheng/Desktop/MACS30000/persp-analysis_A18/Assignments, inifile
plugins: remotedata-0.2.1, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 247 items

test_month_length.py .
test_oper.py .
test_r.py .....

```

.....
.....
test_sf.py .

----- coverage: platform darwin, python 3.6.5-final-0 -----

Name	Stmts	Miss	Cover
get_r.py	24	11	54%
month_length.py	10	0	100%
oper.py	14	0	100%
smallest_factor.py	5	0	100%
test_month_length.py	7	0	100%
test_oper.py	16	0	100%
test_r.py	29	0	100%
test_sf.py	6	0	100%
TOTAL	111	11	90%

The coverage report is attached in the folder. As the test doesn't include all the assertion cases, my function, which includes all the assertion, can't realize full coverage.

3 Watt

See attached pdf.