Assignment 7-Question 1&2

MACS 30000 Delores Tang

1. Unit Testing in Python - Problem 1.

The original function (q1.py) and the test file (test_q1.py) are attached in the files.

In [ ]:
```
# Define Smallest Factor function
def smallest_function():
 """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 [ ]:
```
# Define the test function in test_q1.py
import q1

def test_smallest_factor():
    assert q1.smallest_factor(1) == 1
    assert q1.smallest_factor(2) == 2
    assert q1.smallest_factor(3) == 3
    assert q1.smallest_factor(4) == 2
    assert q1.smallest_factor(5) == 5
    assert q1.smallest_factor(11) == 11
    assert q1.smallest_factor(13) == 13
    assert q1.smallest_factor(1000) == 2
    assert q1.smallest_factor(33) == 3
    assert q1.smallest_factor(829) == 829
```

```
(base) C:\Users\delor\Desktop\Question1>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\delor\Desktop\Question1, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, arraydiff-0.2
collected 1 item

test_q1.py F                                                             [100%]

================================== FAILURES ===================================
_____________________________ test_smallest_factor _____________________________

    def test_smallest_factor():
        assert q1.smallest_factor(1) == 1
        assert q1.smallest_factor(2) == 2
        assert q1.smallest_factor(3) == 3
>       assert q1.smallest_factor(4) == 2
E       assert 4 == 2
E        +  where 4 = <function smallest_factor at 0x000001FE41673E18>(4)
E        +    where <function smallest_factor at 0x000001FE41673E18> = q1.smallest_factor

test_q1.py:12: AssertionError
========================== 1 failed in 0.15 seconds ===========================
```

The test failed for n = 4. The function did not return a smallest factor of 2 for n = 4 because the range(2, int(n**.5)) would give i a range of (2,2). I fixed the function by adding one to the upper bound of the range and modified the q1.py file:

In [1]:
```
# Updated function
def smallest_function():

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

```
(base) C:\Users\delor\Desktop\Question1>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\delor\Desktop\Question1, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, arraydiff-0.2
collected 1 item

test_q1.py .                                                             [100%]

========================== 1 passed in 0.04 seconds ===========================
```

Problem 2

My coverage shows that I reached complete coverage for the first problem after correction. The month_length function (in file q2.py) and the test file (test_q2.py) are attached in the folder.

In [2]:
```
# The original 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 [3]:
```
# The test function
import q2

def test_month_length():
    assert q2.month_length("January", leap_year = False) == 31
    assert q2.month_length("February", leap_year = True) == 29
    assert q2.month_length("February", leap_year = False) == 28
    assert q2.month_length("March", leap_year = False) == 31
    assert q2.month_length("April", leap_year = False) == 30
    assert q2.month_length("May", leap_year = False) == 31
    assert q2.month_length("June", leap_year = False) == 30
    assert q2.month_length("July", leap_year = False) == 31
    assert q2.month_length("August", leap_year = False) == 31
    assert q2.month_length("September", leap_year = False) == 30
    assert q2.month_length("October", leap_year = False) == 31
    assert q2.month_length("November", leap_year = False) == 30
    assert q2.month_length("December", leap_year = False) == 31
```

The coverage test showed complete coverage for the test function I wrote for problem 2.

```
(base) C:\Users\delor\Desktop\Question1>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\delor\Desktop\Question1, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 2 items

test_q1.py .                                                             [ 50%]
test_q2.py .                                                             [100%]

========================== 2 passed in 0.07 seconds ===========================

(base) C:\Users\delor\Desktop\Question1>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\delor\Desktop\Question1, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 2 items

test_q1.py .                                                             [ 50%]
test_q2.py .                                                             [100%]

---------- coverage: platform win32, python 3.7.0-final-0 -----------
Name          Stmts   Miss  Cover
-------------------------------
q1.py             5      0   100%
q2.py            10      0   100%
test_q1.py       12      0   100%
test_q2.py       16      0   100%
-------------------------------
TOTAL            43      0   100%


========================== 2 passed in 0.10 seconds ===========================
```

Problem 3.

The original function (q3.py) and the test file (test_q3.py) are attached in the file.

In [4]:
```
# Original Function
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 [5]:
```
# Test file for problem 3

import q3, pytest

def test_operate():
    assert q3.operate(1,3,"+") == 4
    assert q3.operate(-1,-3,"+") == -4
    assert q3.operate(1,3,"-") == -2
    assert q3.operate(-1,-3,"-") == 2
    assert q3.operate(5,3,"*") == 15
    assert q3.operate(0,5,"*") == 0
    assert q3.operate(5,3,"/") == 5/3

    with pytest.raises(ZeroDivisionError) as err:
        q3.operate(2,0,'/')
    assert err.value.args[0]=="division by zero is undefined"

    with pytest.raises(ValueError) as ValErr:
        q3.operate(2,0,'}')
    assert ValErr.value.args[0]=="oper must be one of '+', '/', '-', or '*'"

    with pytest.raises(TypeError) as type_err:
        q3.operate(2,3,9)
    assert type_err.value.args[0] == "oper must be a string"
```

The pytest and coverage test reports complete coverage:

```
(base) C:\Users\delor\Desktop\Question1>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\delor\Desktop\Question1, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 3 items

test_q1.py .                                                             [ 33%]
test_q2.py .                                                             [ 66%]
test_q3.py .                                                             [100%]

========================== 3 passed in 0.08 seconds ==========================

(base) C:\Users\delor\Desktop\Question1>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\delor\Desktop\Question1, inifile:
plugins: remotedata-0.3.0, openfiles-0.3.0, doctestplus-0.1.3, cov-2.6.0, arraydiff-0.2
collected 3 items

test_q1.py .                                                             [ 33%]
test_q2.py .                                                             [ 66%]
test_q3.py .                                                             [100%]

---------- coverage: platform win32, python 3.7.0-final-0 -----------
Name          Stmts   Miss  Cover
-------------------------------
q1.py             5      0   100%
q2.py            10      0   100%
q3.py            14      0   100%
test_q1.py       12      0   100%
test_q2.py       16      0   100%
test_q3.py       19      0   100%
-------------------------------
TOTAL            76      0   100%


========================== 3 passed in 0.12 seconds ==========================
```

2. Test Driven Development

b) The get_r.py file is attached in the Question2 folder.

In [1]:
```
import numpy as np

def get_r(K, L, alpha, Z, delta):
    "Defining the function for interest rate r_t in a given period of time"

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

    assert alpha >= 0 and alpha <= 1, "Alpha should be within the range (0,1)"
    assert delta >= 0 and delta <= 1, "Delta should be within the range (0,1)"
    assert Z > 0, "Z should be greater than 0"

    if type(K) == float and type(L) == float:
        assert type(r) == float, "Function failed to return scalar interest rate for scalars K and L"
    if not np.isscalar(K) and not np.isscalar(L):
        assert not np.isscalar(r), "Function failed to return vector interest rate for vectors K and L"
    return r
```

The get_r() function passed all tests with complete coverage.

In [ ]: