КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп’ютерних наук та кібернетики

Звіт

Валідація та верифікація програмних систем

Data driven tests

Виконав студент
групи ІНФ-1
Веремчук Максим Романович

Опис інструментарію

В ході виконання лабораторної роботи проводилось тестування простих функцій калькулятора на різних даних декількома способами. Використовувалась мова програмування python в редакторі Visual Studio Code на системі Ubuntu 18.04. Використовувались допоміжні бібліотеки тестування pytest та yaml для представлення даних у зручному вигляді.

Порядок роботи

Було створено 82 тести для верифікації функцій та деякого функціоналу. Було створено клас декоратор задля передачі в тестові функції різні набори даних, що лежить в основі Data driven tests. Наступний код - тести в яких дані нажходять або за допомогою декоратору або ж з yaml файлу:

```
class FooTestCase(unittest.TestCase):
    def test_undecorated(self):
        self.assertTrue(larger_than_two(24))

    @data(3, 4, 12, 23)
    def test_larger_than_two(self, value):
        self.assertTrue(larger_than_two(value))

    @data(1, -3, 2, 0)
    def test_not_larger_than_two(self, value):
        self.assertFalse(larger_than_two(value))

    @data(annotated(2, 1), annotated(10, 5))
    def test_greater(self, value):
        a, b = value
        self.assertGreater(a, b)

    @data(annotated2([2, 1], 'Test_case_1', """Test docstring 1"""),
          annotated2([10, 5], 'Test_case_2', """Test docstring 2"""))
    def test_greater_with_name_docstring(self, value):
        a, b = value
        self.assertGreater(a, b)
        self.assertIsNotNone(getattr(value, "__name__"))
```

```
        self.assertIsNotNone(getattr(value, "__doc__"))

    @file_data('data/test_data_dict_dict.json')
    def test_file_data_json_dict_dict(self, start, end, value):
        self.assertLess(start, end)
        self.assertLess(value, end)
        self.assertGreater(value, start)

    @file_data('data/test_data_dict.json')
    def test_file_data_json_dict(self, value):
        self.assertTrue(has_three_elements(value))

    @file_data('data/test_data_list.json')
    def test_file_data_json_list(self, value):
        self.assertTrue(is_a_greeting(value))

    @needs_yaml
    @file_data('data/test_data_dict_dict.yaml')
    def test_file_data_yaml_dict_dict(self, start, end, value):
        self.assertLess(start, end)
        self.assertLess(value, end)
        self.assertGreater(value, start)

    @needs_yaml
    @file_data('data/test_data_dict.yaml')
    def test_file_data_yaml_dict(self, value):
        self.assertTrue(has_three_elements(value))

    @needs_yaml
    @file_data('data/test_data_list.yaml')
    def test_file_data_yaml_list(self, value):
        self.assertTrue(is_a_greeting(value))

    @data((3, 2), (4, 3), (5, 3))
    @unpack
    def test_tuples_extracted_into_arguments(self, first_value,
second_value):
        self.assertTrue(first_value > second_value)

    @data([3, 2], [4, 3], [5, 3])
    @unpack
    def test_list_extracted_into_arguments(self, first_value,
second_value):
```

```
        self.assertTrue(first_value > second_value)

    @unpack
    @data({'first': 1, 'second': 3, 'third': 2},
          {'first': 4, 'second': 6, 'third': 5})
    def test_dicts_extracted_into_kwargs(self, first, second, third):
        self.assertTrue(first < third < second)

    @data(u'ascii', u'non-ascii-\N{SNOWMAN}')
    def test_unicode(self, value):
        self.assertIn(value, (u'ascii', u'non-ascii-\N{SNOWMAN}'))

    @data(3, 4, 12, 23)
    def test_larger_than_two_with_doc(self, value):
        """Larger than two with value {0}"""
        self.assertTrue(larger_than_two(value))

    @data(3, 4, 12, 23)
    def test_doc_missing_args(self, value):
        """Missing args with value {0} and {1}"""
        self.assertTrue(larger_than_two(value))

    @data(3, 4, 12, 23)
    def test_doc_missing_kargs(self, value):
        """Missing kargs with value {value} {value2}"""
        self.assertTrue(larger_than_two(value))

    @data([3, 2], [4, 3], [5, 3])
    @unpack
    def test_list_extracted_with_doc(self, first_value, second_value):
        """Extract into args with first value {} and second value {}"""
        self.assertTrue(first_value > second_value)
```

Приклад yaml файлу з даними:

```
positive_integer_range:
    start: 0
    end: 2
    value: 1

negative_integer_range:
    start: -2
    end: 0
```

```
   value: -1

positive_real_range:
   start: 0.0
   end: 1.0
   value: 0.5

negative_real_range:
   start: -1.0
   end: 0.0
   value: -0.5
```

або схожий файл у форматі json:

```
{
   "positive_integer_range": {
       "start": 0,
       "end": 2,
       "value": 1
   },
   "negative_integer_range": {
       "start": -2,
       "end": 0,
       "value": -1
   },
   "positive_real_range": {
       "start": 0.0,
       "end": 1.0,
       "value": 0.5
   },
   "negative_real_range": {
       "start": -1.0,
       "end": 0.0,
       "value": -0.5
   }
}
```

Код функціоналу, що здійснює декорування:

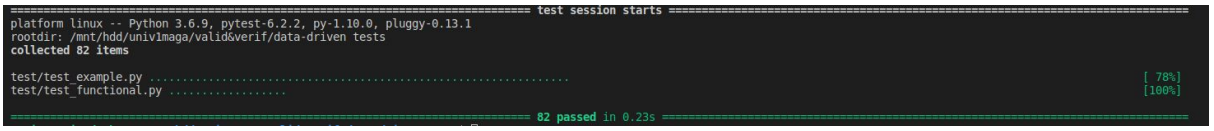
```
def ddt(arg=None, **kwargs):
      fmt_test_name = kwargs.get("testNameFormat",
TestNameFormat.DEFAULT)

   def wrapper(cls):
       for name, func in list(cls.__dict__.items()):
           if hasattr(func, DATA_ATTR):
               for i, v in enumerate(getattr(func, DATA_ATTR)):
                   test_name = mk_test_name(
                       name,
                       getattr(v, "__name__", v),
                       i,
                       fmt_test_name
                   )
                   test_data_docstring = _get_test_data_docstring(func,
v)
                   if hasattr(func, UNPACK_ATTR):
                       if isinstance(v, tuple) or isinstance(v, list):
                           add_test(
                               cls,
                               test_name,
                               test_data_docstring,
                               func,
                               *v
                           )
                       else:
                           # unpack dictionary
                           add_test(
                               cls,
                               test_name,
                               test_data_docstring,
                               func,
                               **v
                           )
                   else:
                       add_test(cls, test_name, test_data_docstring,
func, v)
               delattr(cls, name)
           elif hasattr(func, FILE_ATTR):
               file_attr = getattr(func, FILE_ATTR)
               process_file_data(cls, name, func, file_attr)
               delattr(cls, name)
       return cls
```

```
    return wrapper(arg) if inspect.isclass(arg) else wrapper
```

Результати тестування

Вісі тести відпрацювали без помилок:



Висновки

У ході виконання лабораторної роботи було виконано тестування функцій за допомогою великого масиву даних, які передавались або у вигляді декоратора напряму, або через yaml чи json файли. Усі тести працюють коректно.