Міністерство освіти і науки України

Державний університет “Житомирська політехніка”

Кафедра інженерії програмного забезпечення

Група: ВТ-21-1[1]

Програмування мовою Python

Лабораторна робота № 10

«unit-тестування в мові Python»

Виконав: Бабушко А. С.

Прийняв: Морозов Д. С.

Мета роботи: ознайомитися з фреймворками для unit-тестування в мові Python.

Хід роботи:

Використовуючи фреймворки для unit-тестування unittest або pytest напишіть тести для класів створених під час виконання завданнь 8 і 9 в лабораторній роботі №8. За потреби модифікуйте код класів для виправлення можливих помилок, що будуть знайдені під час покриття класів тестами.

Завдання на лабораторну роботу:

1. Онлайн-магазин (Завдання 8 з Л.Р. №8).
    a. Створіть клас з ім'ям Shop(). Клас Shop() повинен містити два атрибути: shop_name і store_type. Створіть метод describe_shop(), який виводить два атрибути, і метод open_shop(), який виводить повідомлення про те, що онлайн-магазин відкритий. Створіть на основі класу екземпляр з ім'ям store. Виведіть два атрибути окремо, потім викличте обидва методи.
    b. Створіть три різних екземпляри класу, викличте для кожного екземпляру метод describe_shop().
    c. Додайте атрибут number_of_units зі значенням за замовчуванням 0; він представляє кількість видів товару у магазині. Створіть екземпляр з ім'ям store. Виведіть значення number_of_units, а потім змініть number_of_units і виведіть знову.
    d. Додайте метод з ім'ям set_number_of_units(), що дозволяє задати кількість видів товару. Викличте метод з новим числом, знову виведіть значення. Додайте метод з ім'ям increment_number_of_units(), який збільшує кількість видів товару на задану величину. Викличте цей метод.
    e. Напишіть клас Discount(), що успадковує від класу Shop(). Додайте атрибут з ім'ям discount_products для зберігання списку товарів, на які встановлена знижка. Напишіть метод get_discounts_ptoducts, який виводить цей список. Створіть екземпляр store_discount і викличте цей метод.
    f. Збережіть код класу Shop() у модулі. Створіть окремий файл, що імпортує клас Shop(). Створіть екземпляр all_store і викличте один з методів Shop(), щоб перевірити, що команда import працює правильно.

Лістинг програми:

1. shop.py:

```
class Shop:
    def __init__(self, shop_name: str, shop_type: str, shop_number_of_units: int = 0):
        self.name: str = shop_name
        self.type: str = shop_type
        self.number_of_units: int = shop_number_of_units
```

```
    def describe_shop(self) -> str:
        return f'Shop name: {self.name}; Shop type: {self.type};'

    @staticmethod
    def open_shop() -> str:
        return 'Online shop is opened!'

    def set_number_of_units(self, number_of_units: int) -> int:
        if number_of_units >= 0:
            self.number_of_units: int = number_of_units
        return self.number_of_units

    def increment_number_of_units(self, increment: int) -> int:
        if increment > 0:
            self.number_of_units += increment
        return self.number_of_units
```

2. discount.py:

```
from shop import Shop


class Discount(Shop):
    def __init__(self, shop_name: str, shop_type: str, **kwargs) -> None:
        super().__init__(shop_name, shop_type)
        self.discount_products: dict = kwargs

    def get_discounts_products(self) -> None:
        discount_keys: list = list(self.discount_products.keys())
        discount_values: list = list(self.discount_products.values())

        print(f'Current discounts:')
        for iterator in range(0, len(self.discount_products)):
            print(f'{discount_keys[iterator]}: {discount_values[iterator]}')
```

3. lab10.py:

```
""" Lab 10. Python. Andrii Babushko. Repository: https://github.com/AndriiBabushko/Python
"""
import sys
import unittest
from shop import Shop
from discount import Discount
from user import User
from admin import Admin

sys.path.insert(0, r'modules')


# task 1
class ShopDiscountCheck(unittest.TestCase):
    @classmethod
    def setUpClass(cls) -> None:
        print('Set up class')
        print('---------------')

    @classmethod
    def tearDownClass(cls) -> None:
        print('---------------')
        print('Tear down class')

    def setUp(self) -> None:
        print(f'Set up for ["{self.shortDescription()}"]')

    def tearDown(self) -> None:
        print(f'Tear down for ["{self.shortDescription()}"]\n')

    @staticmethod
    def create_shop() -> Shop:
        return Shop('All  store', 'store')
```

```
    @staticmethod
    def create_discount() -> Discount:
        return Discount('Store', 'stuff store', Car_Toy='10%', Doll_toy='20%',
Mobile_phone='15%')

    def test_shop_describe_shop(self):
        shop: Shop = self.create_shop()
        print(f'Shop describe shop id: {self.id()}')
        self.assertMultiLineEqual(shop.describe_shop(), f'Shop name: {shop.name}; Shop
type: {shop.type};')

    def test_shop_open_shop(self):
        shop: Shop = self.create_shop()
        print(f'Shop open shop id: {self.id()}')
        self.assertMultiLineEqual(shop.open_shop(), 'Online shop is opened!')

    def test_shop_set_number_of_units(self):
        shop: Shop = self.create_shop()
        print(f'Shop set number of units id: {self.id()}')
        self.assertEqual(shop.set_number_of_units(5), 5)
        self.assertEqual(shop.set_number_of_units(0), 0)
        self.assertEqual(shop.set_number_of_units(-5), 0)

    def test_shop_increment_number_of_units(self):
        shop: Shop = self.create_shop()
        print(f'Shop increment number of units id: {self.id()}')
        self.assertEqual(shop.increment_number_of_units(5), 5)
        self.assertEqual(shop.increment_number_of_units(-10), 5)
        self.assertEqual(shop.increment_number_of_units(0), 5)

    def test_discount_get_discounts_products(self):
        discount: Discount = self.create_discount()
        print(f'Shop get discounts products id: {self.id()}')
        self.assertIsNone(discount.get_discounts_products())
```

Результат програми:

```
Tests passed: 5 of 5 tests – 9 ms
C:\Users\A\PycharmProjects\pythonProject\University\Scripts\python.exe "D:/PC Programs/PyCharm
Testing started at 21:32 ...
Launching unittests with arguments python -m unittest lab10.ShopDiscountCheck in D:\Політех\2_c

Set up class
--------------
Set up for ["None"]
Shop get discounts products id: lab10.ShopDiscountCheck.test_discount_get_discounts_products
Current discounts:
Car_Toy: 10%
Doll_toy: 20%
Mobile_phone: 15%
Tear down for ["None"]

Set up for ["None"]
Shop describe shop id: lab10.ShopDiscountCheck.test_shop_describe_shop
Tear down for ["None"]

Set up for ["None"]
Shop increment number of units id: lab10.ShopDiscountCheck.test_shop_increment_number_of_units
Tear down for ["None"]

Set up for ["None"]
Shop open shop id: lab10.ShopDiscountCheck.test_shop_open_shop
Tear down for ["None"]

Set up for ["None"]
Shop set number of units id: lab10.ShopDiscountCheck.test_shop_set_number_of_units
Tear down for ["None"]

--------------
Tear down class
```

```
Ran 5 tests in 0.009s

OK

Process finished with exit code 0
```

2. Облік користувачів на сайті (Завдання 9 з Л.Р. №8).

   a. Створіть клас з ім'ям User. Створіть два атрибути first_name і last_name, а потім ще кілька атрибутів, які зазвичай зберігаються у профілі користувача (поштова адреса, нікнейм, що відображається на сайті, згода на розсилку новин з форуму). Напишіть метод describe_user який виводить повне ім'я користувача. Створіть ще один метод greeting_user() для виведення персонального вітання для користувача. Створіть кілька примірників, які представляють різних користувачів. Викличте обидва методи для кожного користувача.

   b. Додайте атрибут login_attempts у клас User. Напишіть метод increment_login_attempts(), що збільшує значення login_attempts на 1. Напишіть інший метод з ім'ям reset_login_attempts(), обнуляє значення login_attempts. Створіть екземпляр класу User і викличте increment_login_attempts() кілька разів. Виведіть значення login_attempts, щоб переконатися у тому, що значення було змінено правильно, а потім викличте reset_login_attempts(). Знову виведіть login_attempts і переконайтеся у тому, що значення обнулилося.

   c. Адміністратор - користувач з повними адміністративними привілеями. Напишіть клас з ім'ям Admin, що успадковує від класу User. Додайте атрибут privileges для зберігання списку рядків виду «Allowed to add message», «Allowed to delete users», «Allowed to ban users» і т. д. Напишіть метод show_privileges() для виведення набору привілеїв адміністратора. Створіть екземпляр Admin і викличте метод.

   d. Напишіть клас Privileges. Клас повинен містити всього один атрибут privileges зі списком, який треба забрати із класу Admin. Водночас, необхідно перемістити метод show_privileges() у клас Privileges із класу Admin. Створіть екземпляр priv як атрибут класу Admin. Створіть новий екземпляр admin і використайте метод для виведення списку привілеїв.

   e. Збережіть клас User в одному модулі, а класи Privileges і Admin у іншому модулі. В окремому файлі створіть екземпляр admin і викличте метод show_privileges(), щоб перевірити, що все працює правильно.

Лістинг програми:

1. ***user.py:***

```
class User:
    def __init__(self, first_name: str, last_name: str, email: str, nickname: str,
mailing_consent: bool, login_attempts: int = 0) -> None:
        self.first_name: str = first_name
        self.last_name: str = last_name
        self.full_name: str = last_name + ' ' + first_name
        self.email: str = email
        self.nickname: str = nickname
        self.mailing_consent: bool = mailing_consent
        self.login_attempts: int = login_attempts
```

```
    def describe_user(self) -> str:
        return f'Full user name: {self.full_name}'

    def greeting_user(self) -> str:
        return f'Our greetings, {self.full_name}!'

    def increment_login_attempts(self) -> int:
        self.login_attempts += 1
        return self.login_attempts

    def reset_login_attempts(self) -> int:
        self.login_attempts = 0
        return self.login_attempts
```

2. ***admin.py:***

```
from privileges import Privileges
from user import User


class Admin(User):
    def __init__(self, first_name: str, last_name: str, email: str, nickname: str,
mailing_consent: bool, admin_privileges: list) -> None:
        super().__init__(first_name, last_name, email, nickname, mailing_consent)
        self.admin_privileges: Privileges = Privileges(admin_privileges)

    def show_privileges(self) -> None:
        print(f'{self.full_name} admin\'s privileges:')
        for privilege in self.admin_privileges.privileges:
            print('* ' + privilege)
```

3. ***privileges.py:***

```
class Privileges:
    def __init__(self, privileges: list) -> None:
        self.privileges = privileges

    def show_privileges(self) -> None:
        print('Privileges:')
        for privilege in self.privileges:
            print('* ' + privilege)
```

4. ***lab10.py:***

```
# task 2
class UserAdminPrivilegesCheck(unittest.TestCase):
    @classmethod
    def setUpClass(cls) -> None:
        print('Set up class')
        print('---------------')

    @classmethod
    def tearDownClass(cls) -> None:
        print('---------------')
        print('Tear down class')

    def setUp(self) -> None:
        print(f'Set up for ["{self.shortDescription()}"]')

    def tearDown(self) -> None:
        print(f'Tear down for ["{self.shortDescription()}"]\n')

    @staticmethod
    def create_user() -> User:
        return User('Andrii', 'Babushko', 'andriibabushko@gmail.com', 'AndriiRaccoon',
True)

    @staticmethod
    def create_admin() -> Admin:
        return Admin('Andrii', 'Babushko', 'andriibabushko@gmail.com', 'AndriiRaccoon',
True,
```

```
                    ['Allowed to block users', 'Allowed to delete messages', 'Allowed to
send voices', 'Allowed to delete users'])

    def test_user_describe_user(self):
        user: User = self.create_user()
        print(f'User describe user id: {self.id()}')
        self.assertMultiLineEqual(user.describe_user(), f'Full user name:
{user.full_name}')

    def test_user_greeting_user(self):
        user: User = self.create_user()
        print(f'User greeting user id: {self.id()}')
        self.assertMultiLineEqual(user.greeting_user(), f'Our greetings,
{user.full_name}!')

    def test_user_increment_login_attempts(self):
        user: User = self.create_user()
        print(f'User increment login attempts id: {self.id()}')
        self.assertEqual(user.increment_login_attempts(), 1)

    def test_user_reset_login_attempts(self):
        user: User = self.create_user()
        print(f'User reset login attempt id: {self.id()}')
        self.assertEqual(user.reset_login_attempts(), 0)

    def test_admin_show_privileges(self):
        admin: Admin = self.create_admin()
        print(f'Admin show privileges id: {self.id()}')
        self.assertIsNone(admin.show_privileges())


if __name__ == '__main__':
    unittest.main()
```

Результат програми:

```
Tests passed: 5 of 5 tests – 5 ms
C:\Users\A\PycharmProjects\pythonProject\University\Scripts\python.exe "D:/PC Programs/PyCharm 2022.
Testing started at 21:35 ...
Launching unittests with arguments python -m unittest lab10.UserAdminPrivilegesCheck in D:\Політех\2

Set up class
--------------
Set up for ["None"]
Admin show privileges id: lab10.UserAdminPrivilegesCheck.test_admin_show_privileges
Babushko Andrii admin's privileges:
* Allowed to block users
* Allowed to delete messages
* Allowed to send voices
* Allowed to delete users
Tear down for ["None"]

Set up for ["None"]
User describe user id: lab10.UserAdminPrivilegesCheck.test_user_describe_user
Tear down for ["None"]

Set up for ["None"]
User greeting user id: lab10.UserAdminPrivilegesCheck.test_user_greeting_user
Tear down for ["None"]

Set up for ["None"]
User increment login attempts id: lab10.UserAdminPrivilegesCheck.test_user_increment_login_attempts
Tear down for ["None"]

Set up for ["None"]
User reset login attempt id: lab10.UserAdminPrivilegesCheck.test_user_reset_login_attempts
Tear down for ["None"]

--------------
Tear down class
```

```
Ran 5 tests in 0.008s

OK

Process finished with exit code 0
```

Увесь лістинг програми:

```
""" Lab 10. Python. Andrii Babushko. Repository: https://github.com/AndriiBabushko/Python
"""
import sys
import unittest
from shop import Shop
from discount import Discount
from user import User
from admin import Admin

sys.path.insert(0, r'modules')


# task 1
class ShopDiscountCheck(unittest.TestCase):
    @classmethod
    def setUpClass(cls) -> None:
        print('Set up class')
        print('---------------')

    @classmethod
    def tearDownClass(cls) -> None:
        print('---------------')
        print('Tear down class')

    def setUp(self) -> None:
        print(f'Set up for ["{self.shortDescription()}"]')

    def tearDown(self) -> None:
        print(f'Tear down for ["{self.shortDescription()}"]\n')

    @staticmethod
    def create_shop() -> Shop:
        return Shop('All  store', 'store')

    @staticmethod
    def create_discount() -> Discount:
        return Discount('Store', 'stuff store', Car_Toy='10%', Doll_toy='20%',
Mobile_phone='15%')

    def test_shop_describe_shop(self):
        shop: Shop = self.create_shop()
        print(f'Shop describe shop id: {self.id()}')
        self.assertMultiLineEqual(shop.describe_shop(), f'Shop name: {shop.name}; Shop
type: {shop.type};')

    def test_shop_open_shop(self):
        shop: Shop = self.create_shop()
        print(f'Shop open shop id: {self.id()}')
        self.assertMultiLineEqual(shop.open_shop(), 'Online shop is opened!')

    def test_shop_set_number_of_units(self):
        shop: Shop = self.create_shop()
        print(f'Shop set number of units id: {self.id()}')
        self.assertEqual(shop.set_number_of_units(5), 5)
        self.assertEqual(shop.set_number_of_units(0), 0)
        self.assertEqual(shop.set_number_of_units(-5), 0)

    def test_shop_increment_number_of_units(self):
        shop: Shop = self.create_shop()
        print(f'Shop increment number of units id: {self.id()}')
        self.assertEqual(shop.increment_number_of_units(5), 5)
        self.assertEqual(shop.increment_number_of_units(-10), 5)
        self.assertEqual(shop.increment_number_of_units(0), 5)

    def test_discount_get_discounts_products(self):
        discount: Discount = self.create_discount()
        print(f'Shop get discounts products id: {self.id()}')
```

```
        self.assertIsNone(discount.get_discounts_products())


# task 2
class UserAdminPrivilegesCheck(unittest.TestCase):
    @classmethod
    def setUpClass(cls) -> None:
        print('Set up class')
        print('---------------')

    @classmethod
    def tearDownClass(cls) -> None:
        print('---------------')
        print('Tear down class')

    def setUp(self) -> None:
        print(f'Set up for ["{self.shortDescription()}"]')

    def tearDown(self) -> None:
        print(f'Tear down for ["{self.shortDescription()}"]\n')

    @staticmethod
    def create_user() -> User:
        return User('Andrii', 'Babushko', 'andriibabushko@gmail.com', 'AndriiRaccoon',
True)

    @staticmethod
    def create_admin() -> Admin:
        return Admin('Andrii', 'Babushko', 'andriibabushko@gmail.com', 'AndriiRaccoon',
True,
                     ['Allowed to block users', 'Allowed to delete messages', 'Allowed to
send voices', 'Allowed to delete users'])

    def test_user_describe_user(self):
        user: User = self.create_user()
        print(f'User describe user id: {self.id()}')
        self.assertMultiLineEqual(user.describe_user(), f'Full user name:
{user.full_name}')

    def test_user_greeting_user(self):
        user: User = self.create_user()
        print(f'User greeting user id: {self.id()}')
        self.assertMultiLineEqual(user.greeting_user(), f'Our greetings,
{user.full_name}!')

    def test_user_increment_login_attempts(self):
        user: User = self.create_user()
        print(f'User increment login attempts id: {self.id()}')
        self.assertEqual(user.increment_login_attempts(), 1)

    def test_user_reset_login_attempts(self):
        user: User = self.create_user()
        print(f'User reset login attempt id: {self.id()}')
        self.assertEqual(user.reset_login_attempts(), 0)

    def test_admin_show_privileges(self):
        admin: Admin = self.create_admin()
        print(f'Admin show privileges id: {self.id()}')
        self.assertIsNone(admin.show_privileges())


if __name__ == '__main__':
    unittest.main()
```

Висновок: під час виконання лабораторної роботи було отримано навички написання власних unit тестів з використанням фреймворків для тестування вже написаного коду у минулих лабораторних роботах.