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).
   1. 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.
   2. Створіть три різних екземпляри класу, викличте для кожного екземпляру метод describe\_shop().
   3. 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.
   4. 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.
   5. 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.
   6. 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

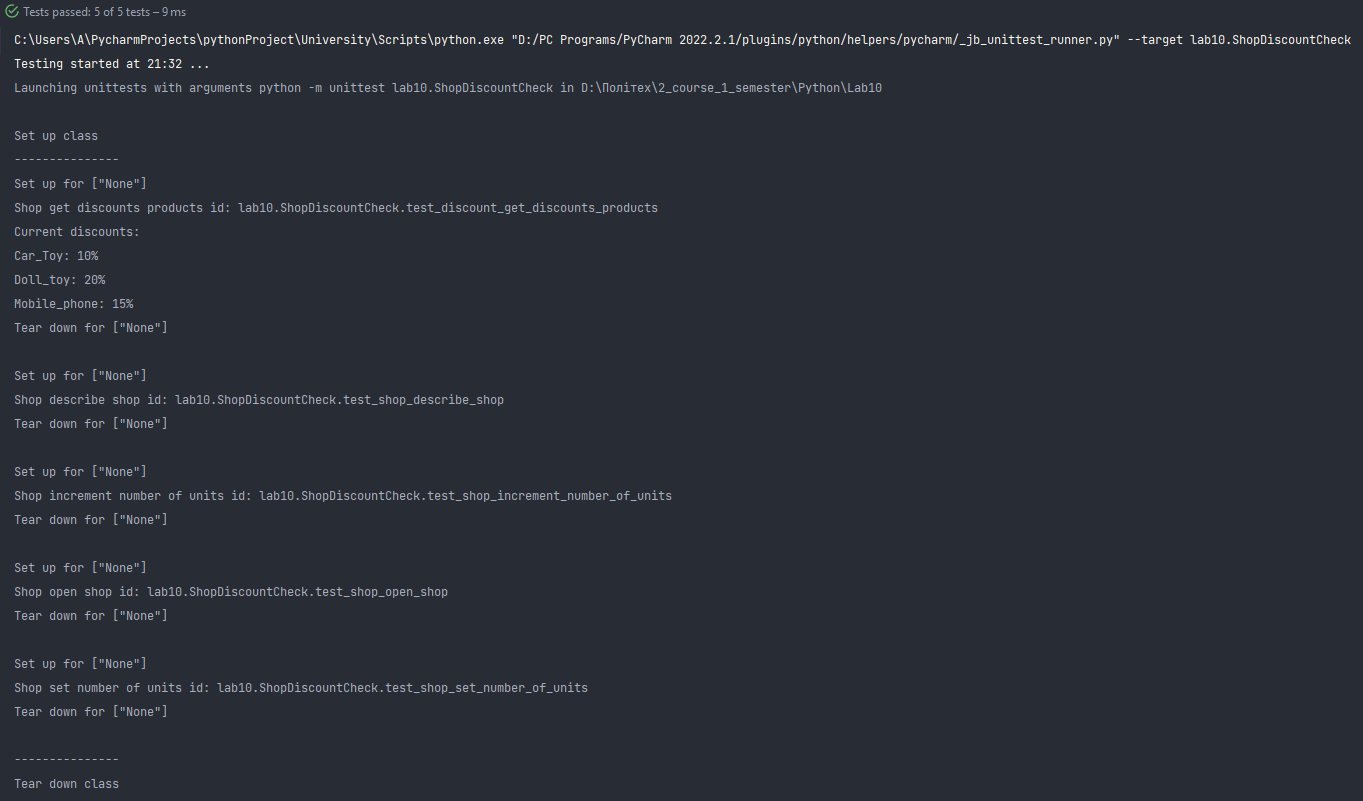
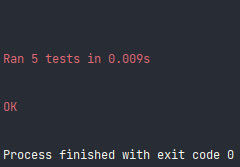
1. ***discount.py:***

*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]}')

1. ***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:***

1. Облік користувачів на сайті (Завдання 9 з Л.Р. №8).
   1. 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.
   2. 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.
   3. 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.
   4. 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.
   5. 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

1. ***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)

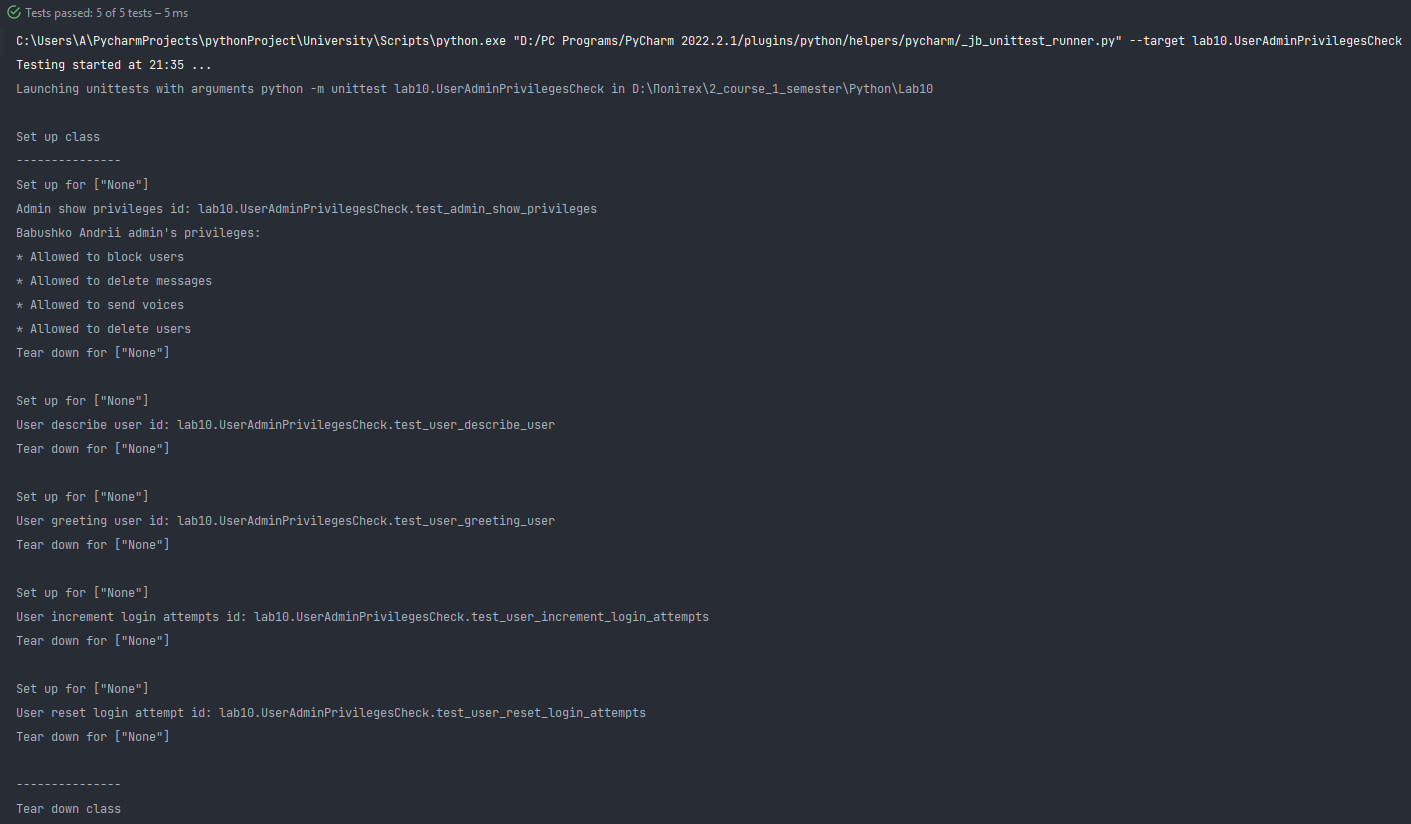
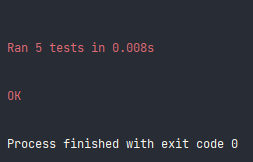
1. ***privileges.py:***

*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)

1. ***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:***

*** ***

***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.