Міністерство освіти і науки України

Державний університет „Житомирська політехніка”

Кафедра КІ та КБ

Група: КБ-21-1

Програмування мовою Python

Лабораторна робота №8

«КЛАСИ. Ч. 2»

Виконала: Поліщук К. Р.

Прийняв: Морозов Д. С.

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.125.10.000 –Лр.8		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Поліщук К.Р.			Звіт з лабораторної роботи №8	Літ.	Арк. 1 / Аркушів 14
Перевір.		Морозов Д. С.					
Керівник						ФІКТ, гр. КБ 21-1-[1]	
Н. контр.							
Затверд.							

Мета роботи: ознайомитися з ООП в мові Python

Хід роботи

Завдання 1: Напишіть клас Bank для опису простих операції з вашим банківським рахунком: покласти на рахунок, зняти з рахунку, переглянути рахунок. При створенні екземпляру класу, екземпляр отримує атрибут __balance з певним значенням. Клас повинен містити методи для додавання коштів на рахунок і знімання з рахунку, за умови, що на рахунку достатньо коштів.

Лістинг програми:

```
def task1():
    class Bank:
        def __init__(self, __balance=0):
            self.__balance = float(__balance)

        def put(self, operation):
            if operation > 0:
                self.__balance += operation
            else:
                return False

        def withdraw(self, operation):
            if self.__balance >= operation > 0:
                self.__balance -= operation
            else:
                return False

        def check(self):
            print(f"Баланс на рахунку => {self.__balance}")

    test = Bank(0)
    test.check()
    test.put(10)
    test.check()
    test.withdraw(5)
    test.check()
    print(test.withdraw(10))
```

Результат виконання програми:

```
------------------------------

Баланс на рахунку => 0.0
Баланс на рахунку => 10.0
Баланс на рахунку => 5.0
False

------------------------------
```

Завдання 2: Напишіть клас Coin, який описує монету, яку можна підкидати. При створенні екземпляру класу, екземпляр отримує атрибут __sideup зі значенням

heads або tails. У класі визначте метод toss, який випадково визначає результат підкидання монети - орел чи решка. Створіть екземпляр класу і виведіть на екран n підкидань монети.

Лістинг програми:

```
import random

def task2():
    class Coin:
        def __init__(self, __sideup='heads'):
            if __sideup != 'heads' or __sideup != 'tails':
                self.__sideup = 'heads'
            self.__sideup = str(__sideup)

        def toss(self):
            if bool(random.getrandbits(1)):
                self.__sideup = 'heads'
                return 'heads'
            else:
                self.__sideup = 'tails'
                return 'tails'

    test = Coin()
    count = 0
    n = int(input('Кількість підкидань -> '))
    for i in range(n):
        if test.toss() == 'heads':
            count += 1

    print(f"З {n} підкидань => {count} 'heads'")
    print(f"З {n} підкидань => {n - count} 'tails'")
```

Результат виконання програми:

```
--------------------------------

Кількість підкидань -> 7777
З 7777 підкидань => 3879 'heads'
З 7777 підкидань => 3898 'tails'
```

Завдання 3: Напишіть клас Car, який надає для створених екземплярів такі атрибути даних автомобіля: марку виготовлення автомобіля, модель автомобіля, рік автомобіля, швидкість (початкове значення 0). Клас також повинен мати наступні методи: accelerate (метод повинен щоразу додавати 5 до значення атрибуту даних про швидкість), brake (метод повинен віднімати 5 від значення атрибута даних швидкості кожного разу, коли він викликається), get_speed (метод повинен повернути поточну швидкість). Створіть екземпляр класу Car і викличте

метод accelerate п’ять разів. Після кожного виклику методу accelerate отримайте поточну швидкість автомобіля і надрукуйте її значення. Потім викличте метод brake п’ять разів. Після кожного виклику методу brake отримайте поточну швидкість автомобіля та надрукуйте її значення.

Лістинг програми:

```
def task3():
    class Car:
        def __init__(self, brand='', model='', year=0):
            self.speed = 0
            self.brand = brand
            self.model = model
            self.year = year

        def accelerate(self):
            self.speed += 5

        def brake(self):
            if self.speed != 0:
                self.speed -= 5
            else:
                return False

        def get_speed(self):
            print(f"Швидкість => {self.speed}")

    test = Car()
    for i in range(5):
        test.accelerate()
        test.get_speed()
    for i in range(5):
        test.brake()
        test.get_speed()
```

Результат виконання програми:

```
Швидкість => 5
Швидкість => 10
Швидкість => 15
Швидкість => 20
Швидкість => 25
Швидкість => 20
Швидкість => 15
Швидкість => 10
Швидкість => 5
Швидкість => 0
```

Завдання 4: Напишіть клас Dog, який має три атрибути класу: mammal (ссавець), nature (характер) і breed (порода), та два атрибути екземпляра: name (кличка) і age (вік). Створіть екземпляри трьох нових собак, кожна з яких різного віку. Визначте у класі Dog метод для виведення значень атрибутів екземпляру - імені та віку конкретної собаки. За потреби, додайте кілька інших методів, які визначають поведінку собаки (подавання голосу тощо). Напишіть кілька класів, які унаслідуються від батьківського класу Dog, що описують конкретні породи собак. Визначте для цих класів атрибути nature і breed відповідно, включіть у класи по одному методу, що визначає поведінку конкретної породи собаки. Створіть батьківський клас Pets, що створює список ваших домашніх улюбленців. У підсумку, надрукуйте інформацію про ваших домашніх тварин, на зразок, як у вихідних даних.

Лістинг програми:

```
def task4():
    class Pets:
        dogs = []

        def __init__(self, dogs):
            self.dogs = dogs

        def info(self):
            for dog in self.dogs:
                print(dog)

    class Dog:
        mammal = True
        nature = str
        breed = str

        def __init__(self, name, age): # атрибути екземпляру
            self.name = name
            self.age = age

        def __str__(self):
            return f"Dog's name is {self.name}, age is {self.age}, breed is
{self.breed} and nature is {self.nature}"

        def voice(self):
            print('Gav')

        def jump(self):
            print('Jump')

    class Husky(Dog):
        nature = 'friendly'
        breed = 'Husky'
```

```
        def __init__(self, name, age):
            super().__init__(name, age)

        def voice(self):
            print('Gav-gav')

    class Pug(Dog):
        nature = 'lazy'
        breed = 'Pug'

        def __init__(self, name, age):
            super().__init__(name, age)

        def voice(self):
            print('Gav-gav-gav')

    class Bulldog(Dog):
        nature = 'aggressive'
        breed = 'Bulldog'

        def __init__(self, name, age):
            super().__init__(name, age)

        def voice(self):
            print('Gav-gav-gav-gav')

    dog1 = Husky('Bob', 5)
    dog2 = Pug('Jack', 3)
    dog3 = Bulldog('Tom', 7)
    dogs = Pets([dog1, dog2, dog3])
    dogs.info()
    dog1.voice()
```

Результат виконання програми:

```
------------------------------

Dog's name is Bob, age is 5, breed is Husky and nature is friendly
Dog's name is Jack, age is 3, breed is Pug and nature is lazy
Dog's name is Tom, age is 7, breed is Bulldog and nature is aggressive
Gav-gav

------------------------------
```

Завдання 5: Дано послідовність цілих чисел. Необхідно її обробити і вивести на екран суму першої п’ятірки чисел із цієї послідовності, потім суму другої п’ятірки, і т. д. Але послідовність не дається відразу загалом. З плином часу до вас надходять її послідовні частини. Наприклад, спочатку перші три елементи, потім наступні шість, потім наступні два і т. д. Реалізуйте клас Buffer, який буде накопичувати в собі елементи послідовності і виводити суму п’ятірок послідовних елементів у міру їх накопичення. Однією з вимог до класу є те, що він не повинен

зберігати в собі більше елементів, ніж йому дійсно необхідно, тобто, він не повинен зберігати елементи, які вже увійшли в п’ятірку, для якої була виведена сума. Клас повинен мати наступний вигляд

class Buffer:

def __init__(self):

конструктор без аргументів

def add(self, *a):

додати наступну частину послідовності

def get_current_part(self):

повернути збережені в поточний момент часу елементи послідовності в порядку, в якому вони були додані

Зверніть увагу, що під час виконання методу add виводити суму п’ятірок може знадобитися кілька разів до тих пір, поки в буфері не залишиться менше п’яти елементів.

Лістинг програми:

```
def task5():
    class Buffer:
        def __init__(self):
            self.part = []

        def add(self, *a):
            a = [val for val in a if isinstance(val, int)]
            self.part.extend(a)
            while len(self.part) >= 5:
                print(f"{self.part[:5]} -> {sum(self.part[:5])}")
                self.part = self.part[5:]

        def get_current_part(self):
            return self.part

    test = Buffer()
    test.add(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
    test.add('hello', 1, 2, 3)
    print(f"Saved -> {test.get_current_part()}")
    test.add(1, 2)
```

Результат виконання програми:

```
------------------------------

[1, 2, 3, 4, 5] -> 15
[6, 7, 8, 9, 10] -> 40
Saved -> [1, 2, 3]
[1, 2, 3, 1, 2] -> 9

------------------------------
```

Завдання 6: Напишіть клас-виняток, на основі вбудованого в Python класу ValueError(). Клас буде представляти перевірку певного імені на основі його довжини. Якщо довжина введеного імені є меншою 10, то має генеруватися виняток як у вихідних даних. У інших випадках нічого не виводиться

Лістинг програми:

```python
def task6():
    class StringError:
        @staticmethod
        def checkName(name):
            if len(name) >= 10:
                raise StringError('Name is too long')

    StringError.checkName('12345')
    StringError.checkName('12345678910')
```

Результат виконання програми:

```
  File "C:\Users\vika7\OneDrive\Рабочий стол\2 course 1 semester\python\lab 8\main.py", line 194, in task6
    StringError.checkName('1234567890')
  File "C:\Users\vika7\OneDrive\Рабочий стол\2 course 1 semester\python\lab 8\main.py", line 191, in checkName
    raise StringError('Name is too long')
TypeError: StringError() takes no arguments
```

Завдання 7: Напишіть один клас для перетворення десяткового числа на число в римській системі числення. І ще один клас для перетворення числа з римської системи числення у десяткове число.

Лістинг програми:

```python
def task7():
    class Convert:
        @staticmethod
        def toRoman(number):
            value = [1, 4, 5, 9, 10, 40, 50, 90, 100, 400, 500, 900, 1000]
            symbol = ["I", "IV", "V", "IX", "X", "XL", "L", "XC", "C", "CD", "D",
"CM", "M"]
            i = 12
            str = ''

            while number:
                count = number // value[i]
```

```
                number %= value[i]
                while count:
                    str += symbol[i]
                    count -= 1
                i -= 1
            return str

        @staticmethod
        def toInt(number):
            symbol = {'I': 1, 'IV': 4, 'V': 5, 'IX': 9, 'X': 10, 'XL': 40, 'L':
50,
                      'XC': 90, 'C': 100, 'CD': 400, 'D': 500, 'CM': 900, 'M':
1000}
            i = 0
            num = 0

            while len(number) > i:
                if len(number) > i + 1 and number[i:i + 2] in symbol:
                    num += symbol[number[i:i + 2]]
                    i += 2
                else:
                    num += symbol[number[i]]
                    i += 1
            return num

    print(f"=> {Convert.toRoman(int(input('Введіть арабське число -> ')))}")
    print(f"=> {Convert.toInt(str(input('Введіть римське число -> ')))}")
```

Результат виконання програми:

```
------------------------------

Введіть арабське число -> 11
=> XI
Введіть римське число -> XI
=> 11

------------------------------
```

Завдання 8: Онлайн-магазин.

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:

ClassList_task8.py:

```
class Shop(object):
    open = False

    def __init__(self, shop_name, store_type=[], number_of_units=0):
        self.shop_name = shop_name
        self.store_type = store_type
        self.number_of_units = number_of_units

    def describe_shop(self):
        print(f"Магазин {self.shop_name} продає {self.store_type}")

    def open_shop(self):
        self.open = True

    def set_number_of_units(self, number_of_units):
        self.number_of_units = number_of_units
        return self.number_of_units

    def increment_number_of_units(self, number_of_units):
        self.number_of_units += number_of_units
        return self.number_of_units


class Discount(Shop):
    def __init__(self, shop_name, discount_products=[]):
        super().__init__(shop_name)
```

```
        self.discount_products = discount_products

    def get_discounts_ptoducts(self):
        print(f"Товари зі знижкою => {self.discount_products}")
```

main.py

```
def task8():
    from ClassList_task8 import Shop, Discount
    print("a)")
    store = Shop('Ювелірні вироби', 'ювелірка')
    print(f"Назва => {store.shop_name}")
    print(f"Тип => {store.store_type}")
    store.open_shop()
    store.describe_shop()

    print("b)")
    store1 = Shop('FS', 'свіжі фрукти')
    store2 = Shop('Tit', 'годинники')
    store3 = Shop('APRICITY', 'стильне взуття')
    store1.describe_shop()
    store2.describe_shop()
    store3.describe_shop()

    print("c)")
    store = Shop('Minimal', 'ювелірні вироби', 5)
    print(f"Кількість видів => {store.number_of_units}")
    store.number_of_units = 10
    print(f"Кількість видів => {store.number_of_units}")

    print("d)")
    print(f"Кількість видів => {store.set_number_of_units(15)}")
    print(f"Кількість видів => {store.increment_number_of_units(5)}")

    print("e)")
    store_discount = Discount(store.shop_name, ['ювелірні вироби'])
    store_discount.get_discounts_ptoducts()

    print("f)")
    all_store = [store, store1, store2, store3]
    all_store[0].describe_shop()
```

Результат виконання програми:

```
a)
Назва => Ювелірні вироби
Тип => ювелірка
Магазин Ювелірні вироби продає ювелірка
b)
Магазин FS продає свіжі фрукти
Магазин Tit продає годинники
Магазин APRICITY продає стильне взуття
c)
Кількість видів => 5
Кількість видів => 10
d)
Кількість видів => 15
Кількість видів => 20
e)
Товари зі знижкою => ['ювелірні вироби']
f)
Магазин Minimal продає ювелірні вироби
```

Завдання 9: Облік користувачів на сайті.

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:

main.py

```
def task9():
    from User_task9 import User
    from Admin_task9 import Admin, Privileges
    print("a)")
    user_1 = User('Karyna', 'Polishchuk', 'krnplschk14@gmail.com', 'krn67', False)
    user_1.describe_user()
    user_1.greeting_user()
    user_2 = User('Карина', 'Поліщук', 'kb211_pkr@student.ztu.edu.ua', 'Karyna
Polishchuk', True)
    user_2.describe_user()
    user_2.greeting_user()
    print("b)")
    print(f"Кількість користувачів {User.login_attempts}")
    print(f"Скидання {User.reset_login_attempts()}")
    print("c), d)")
    priv = Privileges(['can add post', 'can delete post', 'can ban user'])
    admin = Admin('Karyna', 'Polishchuk', 'krnplschk14@gmail.com', 'krn67', True,
priv.privileges)
    priv.show_privileges()
    print("e)")
    admin.show_privileges()
```

User_task9.py

```
class User:
    login_attempts = 0

    def __init__(self, first_name, last_name, email, nickname, mailing_list):
```

```
        self.first_name = first_name
        self.last_name = last_name
        self.email = email
        self.nickname = nickname
        self.mailing_list = mailing_list
        self.increment_login_attempts()

    def describe_user(self):
        print(f"Повне ім'я => {self.last_name} {self.first_name}")

    def increment_login_attempts(self):
        User.login_attempts += 1
        return User.login_attempts

    @classmethod
    def reset_login_attempts(self):
        User.login_attempts = 0
        return User.login_attempts
    greeting_user = lambda self: print(f"Привіт, {self.nickname}!")
```

Admin_task9.py

```
from User_task9 import User

class Privileges:
     def __init__(self, privileges):
        self.privileges = privileges
     def show_privileges(self):
        print(f"Привілеї => {self.privileges}")

class Admin(User):
     def __init__(self, first_name, last_name, email, nickname, mailing_list,
privileges):
         super().__init__(first_name, last_name, email, nickname, mailing_list)
         self.privileges = privileges
     def show_privileges(self):
        print(f"Привілеї => {self.privileges}")
```

Результат виконання програми:

```
a)
Повне ім'я => Polishchuk Karyna
Привіт, krn67!
Повне ім'я => Поліщук Карина
Привіт, Karyna Polishchuk!
b)
Кількість користувачів 2
Скидання 0
c), d)
Привілеї => ['can add post', 'can delete post', 'can ban user']
e)
Привілеї => ['can add post', 'can delete post', 'can ban user']
```

Висновок: у ході виконання лабораторної роботи ми ознайомилися з ООП в мові Python.