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.

**Мета роботи:** ознайомитися з ООП в мові Python

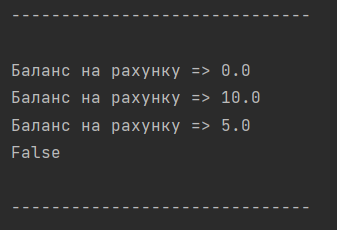
**Хід роботи**

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:



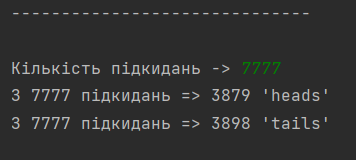
Завдання 2: Напишіть клас Coin, який описує монету, яку можна підкидати. При створенні екземпляру класу, екземпляр отримує атрибут \_\_sideup зі значенням heads або tails. У класі визначте метод toss, який випадково визначає результат підкидання монети - орел чи решка. Створіть екземпляр класу і виведіть 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:**

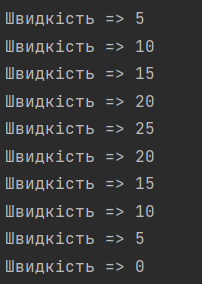


Завдання 3: Напишіть клас Car, який надає для створених екземплярів такі атрибути даних автомобіля: марку виготовлення автомобіля, модель автомобіля, рік автомобіля, швидкість (початкове значення 0). Клас також повинен мати наступні методи: accelerate (метод повинен щоразу додавати 5 до значення атрибуту 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:

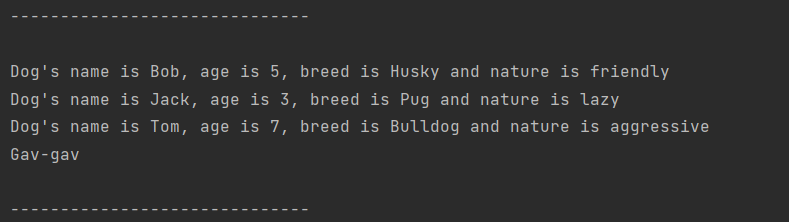


Завдання 4: Напишіть клас Dog, який має три атрибути класу: mammal (ссавець), nature (характер) і breed (порода), та два атрибути ексземпляра: name (кличка) і age (вік). Створіть екземпляри трьох нових собак, кожна з яких різного віку. Визначте у класі Dog метод для виведення значень атрибутів екземпляру - 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:



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):

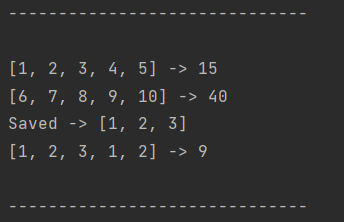
# повернути збережені в поточний момент часу елементи послідовності в порядку, в якому вони були додані

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:

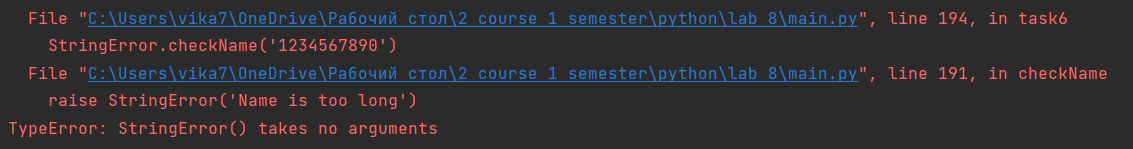


Завдання 6: Напишіть клас-виняток, на основі вбудованого в Python класу ValueError(). Клас буде представляти перевірку певного імені на основі його довжини. Якщо довжина введеного імені є меншою 10, то має генеруватися виняток як у вихідних даних. У інших випадках нічого не виводиться

Лістинг програми:

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:

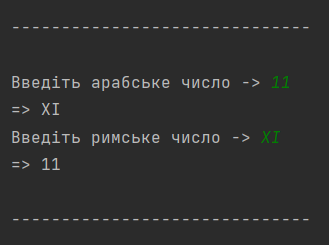


Завдання 7: Напишіть один клас для перетворення десяткового числа на число в римській системі числення. І ще один клас для перетворення числа з римської системи числення у десяткове число.

Лістинг програми:

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:



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:

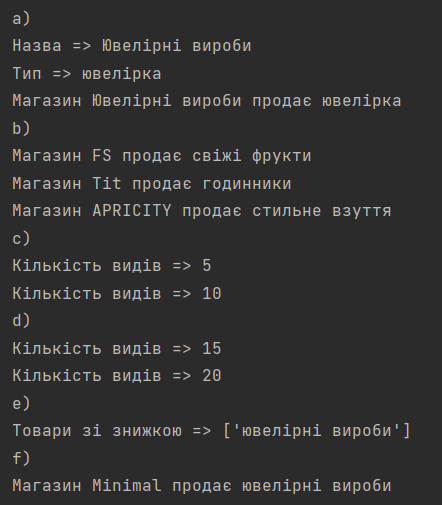
ClassList\_task8.py:

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:



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()

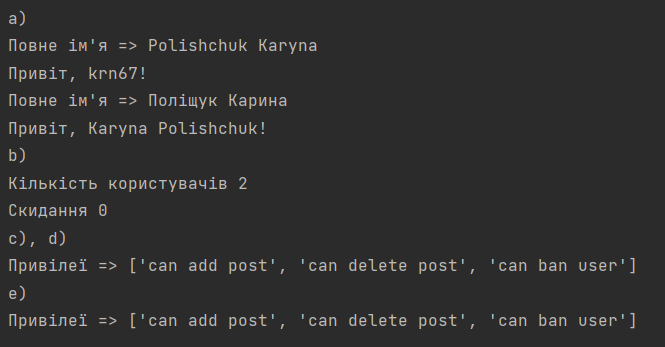
User\_task9.py

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:



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