Державний торговельно-економічний університет  
Кафедра інженерії програмного забезпечення та кібербезпеки

**ЛАБОРАТОРНА РОБОТА №5**

**З ДИСЦИПЛІНИ «АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ»**

**НА ТЕМУ «ООП У МОВІ ПРОГРАМУВАННЯ PYTHON»**

**Виконала**: студентка факультету

інформаційних технологій

групи\_курсу 3-4

Авєріна Наталія Ігорівна

**Перевірила**: Палагута К. О.

Київ 2024

**Лабораторна робота №5**

**Тема**: ООП у мові програмування Python

**Мета**: навчитись працювати з класами, об’єктами, методами, познайомитись з реалізацією механізмів наслідування, інкапсуляції, поліморфізму в мові Python.

**Програмне забезпечення**: PyCharm, MS Word.

**1 варіант**

**Завдання:**

Створити програму, під час виконання якої потрібно:

* реалізувати клас Person, об’єкти якого містять поля – ім’я, вік, адреса, у класі повинні міститися метод додавання поля факультет, метод класу, за допомогою якого можна за датою народження визначити вік, властивості гетер і сетер для роботи з полем вік (контроль коректності даних);
* реалізувати клас Student, що наслідує клас Person, у якому містяться метод \_\_str\_\_, метод для додавання поля група, змінна для обчислення кількості екземплярів класу;
* реалізувати клас Teacher, що наслідує клас Person, у якому містяться метод \_\_str\_\_, методи для додавання полів кафедра, посада, змінна для обчислення кількості екземплярів класу;
* організувати діалог для введення даних про довільну кількість осіб, відомості про конкретну особу зберігати як об’єкти класу Student або Teacher з відповідними полями, кожний об’єкт повинен бути елементом списку;
* роздрукувати список, у якому зберігаються введені дані про осіб, загальну кількість викладачів і студентів.

**Хід роботи:**

1. Створюємо клас Person.
   1. Створюємо метод \_\_init\_\_, який ініціалізує об'єкт з ім'ям, віком та адресою.
   2. Далі визначаємо властивості get та set для роботи з полем віку (перевірка правильності вводу).

class Person:  
 total\_persons = 0  
  
 def \_\_init\_\_(self, name, age, address):  
 self.name = name  
 self.\_age = age  
 self.address = address  
 self.faculty = None  
 Person.total\_persons += 1  
  
 def set\_age(self, age):  
 if age < 0:  
 print("Age cannot be negative.")  
 else:  
 self.\_age = age  
  
 def get\_age(self):  
 return self.\_age  
  
 age = property(get\_age, set\_age)

1. Створюємо клас Student, що наслідує клас Person
   1. Створюємо метод \_\_init\_\_, який ініціалізує об'єкт з додатковим полем group.
   2. Реалізуємо метод \_\_str\_\_, який повертає рядок з інформацією про студента.

class Student(Person):  
 total\_students = 0  
  
 def \_\_init\_\_(self, name, age, address, group):  
 super().\_\_init\_\_(name, age, address)  
 self.group = group  
 Student.total\_students += 1  
  
 def \_\_str\_\_(self):  
 return f"Student: {self.name}, Age: {self.age}, Address: {self.address}, Group: {self.group}"

1. Аналогічно класу Student, клас Teacher успадковує клас Person і містить поля department та position.

class Teacher(Person):  
 total\_teachers = 0  
  
 def \_\_init\_\_(self, name, age, address, department, position):  
 super().\_\_init\_\_(name, age, address)  
 self.department = department  
 self.position = position  
 Teacher.total\_teachers += 1  
  
 def \_\_str\_\_(self):  
 return f"Teacher: {self.name}, Age: {self.age}, Address: {self.address}, Department: {self.department}, Position: {self.position}"

1. Створюємо функцію add\_person()
   1. Завдяки цій функції програма запитує у користувача його особу (студент чи викладач), а потім запитує дані відповідно до класу.
   2. На основі введених даних функція створює відповідний об'єкт студента або викладача і повертає його.

def add\_person():  
 person\_type = input("Enter person type (Student/Teacher): ").lower()  
 name = input("Enter name: ")  
 age = int(input("Enter age: "))  
 address = input("Enter address: ")  
  
 if person\_type == "student":  
 group = input("Enter group: ")  
 return Student(name, age, address, group)  
 elif person\_type == "teacher":  
 department = input("Enter department: ")  
 position = input("Enter position: ")  
 return Teacher(name, age, address, department, position)  
 else:  
 print("Invalid person type.")  
 return None

1. Функція print\_persons() приймає список осіб та друкує їхні дані разом зі загальною кількістю студентів і викладачів.

def print\_persons(persons):  
 print("\nList of Persons:")  
 for person in persons:  
 print(person)  
 print(f"\nTotal Students: {Student.total\_students}")  
 print(f"Total Teachers: {Teacher.total\_teachers}")

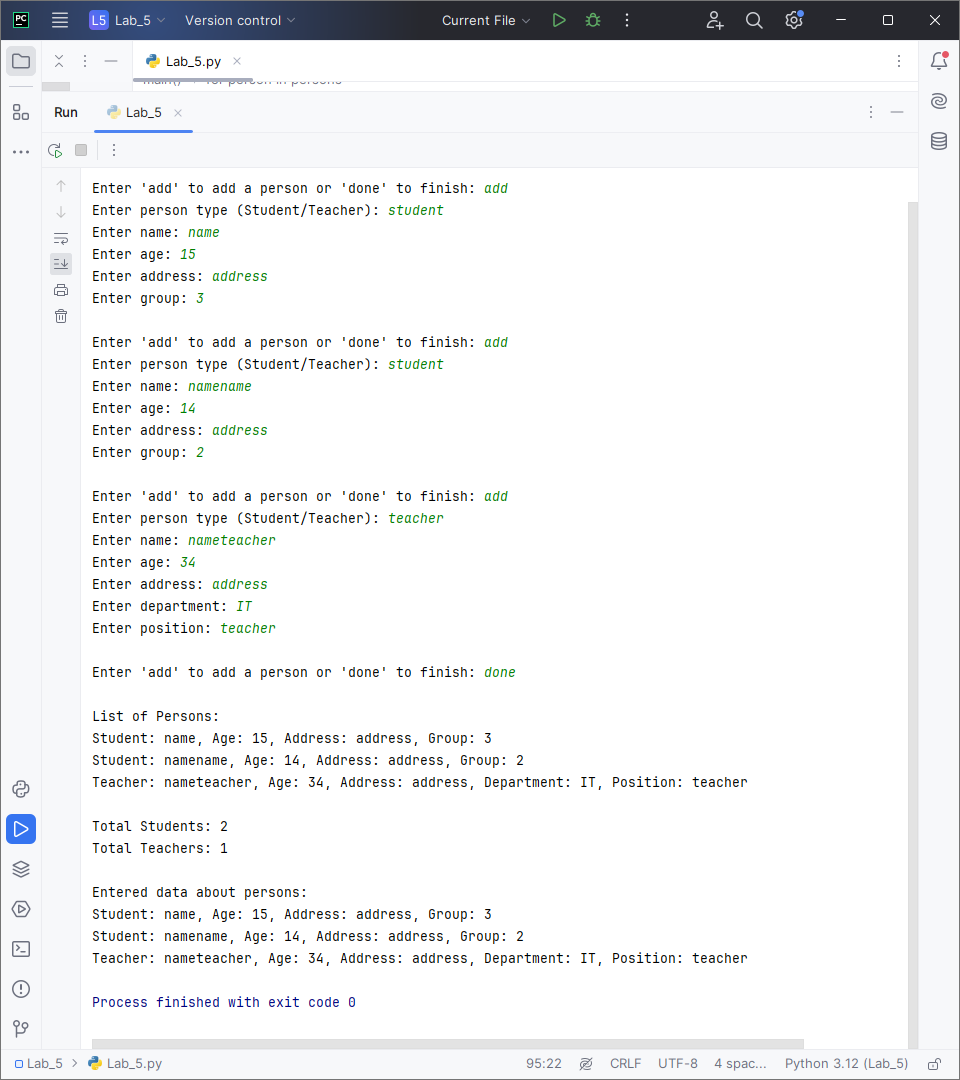
1. Головна функція main() створює список persons. Після цього відбувається виклик функції add\_person() для додавання осіб до списку до тих пір, поки користувач не вибере опцію "done". Після завершення введення даних викликається функція print\_persons() для виведення списку осіб.

def main():  
 persons = []  
  
 while True:  
 choice = input("\nEnter 'add' to add a person or 'done' to finish: ").lower()  
 if choice == "add":  
 person = add\_person()  
 if person:  
 persons.append(person)  
 elif choice == "done":  
 break  
 else:  
 print("Invalid choice.")  
  
 print\_persons(persons)  
 print("\nEntered data about persons:")  
 for person in persons:  
 print(person)

1. Після всього відбувається виклик головної функції.

if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Результат виконання коду:



**Висновки**: в ході виконання практичної роботи було набуто практичних навичок роботи з класами, об’єктами та методами. Було використано механізми наслідування, інкапсуляції та поліморфізму.