Міністерство освіти і науки України

Державний університет ,,Житомирська політехніка”

Кафедра ІПЗ

Група: ІПЗ-23-1

Програмування мовою Python

Лабораторна робота №7

«КЛАСИ. Ч. 1»

Виконав: Риженко Я.В.

Прийняв: Желізко В. В.

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

**Хід роботи**

**Завдання 1:** Реалізувати клас Person, який відображає запис в книзі контактів.

Клас має 4 атрибута:

- surname - рядок - прізвище контакту (обов'язковий)

- first\_name - рядок - ім'я контакту (обов'язковий)

- nickname - рядок - псевдонім (опціональний)

- birth\_date - об'єкт datetime.date (обов'язковий)

Кожен виклик класу повинен створювати екземпляр (інстанс) класу із

зазначеними атрибутами.

Також клас має 2 методи:

- get\_age() - рахує вік особи в повних роках на дату виклику і повертає рядок

виду: "25";

- get\_fullname() - повертає рядок, що відображає повне ім'я (прізвище + ім'я)

контакту;

Примітка:

при створенні атрибута birth\_date з рядка типу "2002-12-31" необхідно:

- визначити яка інформація потрібна для створення об'єкта datetime.date,

- отримати ці дані з рядка

- розібрати її (дістати з неї окремо, рік, місяць, число),

- на підставі цієї інформації створити спеціальний об'єкт datetime.date,

- помістити цей спец.об'ект в атрибут екземпляра класу

**Завдання 2.** Написати функцію modifier(filename), яка приймає ім'я файлу і

повинна:

- прочитати дані з переданого файлу;

- створити об'єкти класу Person на підставі отриманих даних;

- модифікувати дані в файлі:

а) додати графу повного імені (fullname) після графи з ім'ям (name)

б) додати графу з віком (age) в кінець.

На виході отримати файл, розширений зазначеним чином.

**Листинг програми:**

import csv  
from datetime import date  
  
# Task 1  
class Person:  
 def \_\_init\_\_(self, surname: str, first\_name: str, birth\_date: str, nickname: str = None):  
 if not surname.strip():  
 raise ValueError("Прізвище не може бути порожнім.")  
 if not first\_name.strip():  
 raise ValueError("Ім'я не може бути порожнім.")  
  
 self.surname = surname.strip()  
 self.first\_name = first\_name.strip()  
 self.nickname = nickname.strip() if nickname else None  
  
 year, month, day = map(int, birth\_date.split('-'))  
 birth\_date\_obj = date(year, month, day)  
  
 if birth\_date\_obj > date.today():  
 raise ValueError("Дата народження не може бути в майбутньому.")  
  
 self.birth\_date = birth\_date\_obj  
  
 def get\_age(self) -> str:  
 today = date.today()  
 age = today.year - self.birth\_date.year - (  
 (today.month, today.day) < (self.birth\_date.month, self.birth\_date.day)  
 )  
 return str(age)  
  
 def get\_fullname(self) -> str:  
 return f"{self.surname} {self.first\_name}"  
  
  
def create\_person(filename):  
 print("Введіть дані для створення нового контакту.")  
 surname = input("Прізвище: ").strip()  
 first\_name = input("Ім'я: ").strip()  
 birth\_date = input("Дата народження (у форматі YYYY-MM-DD): ").strip()  
 nickname = input("Псевдонім (необов'язково): ").strip() or None  
  
 try:  
 person = Person(surname=surname, first\_name=first\_name, birth\_date=birth\_date, nickname=nickname)  
 print("\nКонтакт успішно створено!")  
 print(f"Повне ім'я: {person.get\_fullname()}")  
 print(f"Вік: {person.get\_age()} років")  
 if person.nickname:  
 print(f"Псевдонім: {person.nickname}")  
  
 with open(filename, 'a', newline='', encoding='utf-8') as file:  
 writer = csv.DictWriter(file, fieldnames=["surname", "name", "birth\_date","nickname"])  
  
 if file.tell() == 0:  
 writer.writeheader()  
  
 writer.writerow({  
 "surname": person.surname,  
 "name": person.first\_name,  
 "birth\_date": person.birth\_date,  
 "nickname": person.nickname or ''  
 })  
  
 print("Контакт успішно додано до файлу.")  
 except ValueError as e:  
 print(f"Помилка: {e}")  
 except Exception as e:  
 print(f"Невідома помилка: {e}")

#Task 2  
def modifier(filename):  
 try:  
 with open(filename, 'r', encoding='utf-8') as file:  
 reader = csv.DictReader(file)  
 fieldnames = reader.fieldnames  
  
 if not {'surname', 'name', 'birth\_date'}.issubset(set(fieldnames)):  
 raise ValueError("Файл повинен містити графи 'surname', 'name', 'birth\_date'.")  
  
 extended\_fieldnames = fieldnames.copy()  
 if 'fullname' not in extended\_fieldnames:  
 extended\_fieldnames.insert(extended\_fieldnames.index('name') + 1, 'fullname')  
 if 'age' not in extended\_fieldnames:  
 extended\_fieldnames.append('age')  
  
 modified\_rows = []  
 for row in reader:  
 surname = row['surname'].strip()  
 name = row['name'].strip()  
 birth\_date = row['birth\_date'].strip()  
  
 person = Person(surname, name, birth\_date)  
  
 row['fullname'] = person.get\_fullname()  
 row['age'] = person.get\_age()  
  
 modified\_rows.append(row)  
  
 with open(filename, 'w', encoding='utf-8', newline='') as file:  
 writer = csv.DictWriter(file, fieldnames=extended\_fieldnames)  
 writer.writeheader()  
 writer.writerows(modified\_rows)  
  
 print("Файл успішно модифіковано.")  
 except FileNotFoundError:  
 print(f"Помилка: Файл {filename} не знайдено.")  
 except ValueError as e:  
 print(f"Помилка: {e}")  
 except Exception as e:  
 print(f"Невідома помилка: {e}")  
  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 create\_person('contacts.csv')  
if \_\_name\_\_ == "\_\_main\_\_":  
 modifier('contacts.csv')

**Результат виконання:**

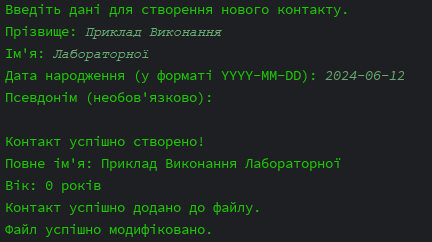
****

Рис. 1. Результат.

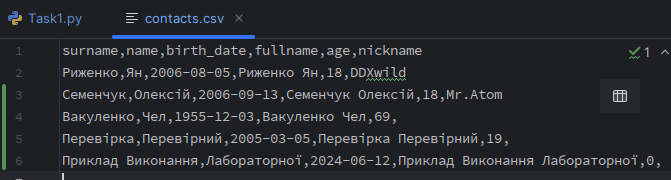


Рис. 2. Створений та модифікований файл.

**Висновок:** У цій лабораторній роботі ми ознайомились з основами ООП мови Python.

***Посилання на Git****: :* *https://github.com/JanRizhenko/Python\_Labs*