Міністерство освіти і науки України

Державний університет “Житомирська політехніка”

Кафедра інженерії програмного забезпечення

Група: ВТ-21-1[1]

Програмування мовою Python

Лабораторна робота № 7

«КЛАСИ. Ч. 1»

Виконав: Бабушко А. С.

Прийняв: Морозов Д. С.

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

***Хід роботи:***

***Завдання на лабораторну роботу:***

Завдання 1: Реалізувати клас Person, який відображає запис в книзі контактів. Клас має 4 атрибута:

1. surname - рядок - прізвище контакту (обов'язковий)
2. first\_name - рядок - ім'я контакту (обов'язковий)
3. nickname - рядок - псевдонім (опціональний)
4. birth\_date - об'єкт datetime.date (обов'язковий)

Кожен виклик класу повинен створювати екземпляр (інстанс) класу із зазначеними атрибутами. Також клас має 2 методи:

1. get\_age() - рахує вік особи в повних роках на дату виклику і повертає рядок виду: "25";
2. get\_fullname() - повертає рядок, що відображає повне ім'я (прізвище + ім'я) контакту;

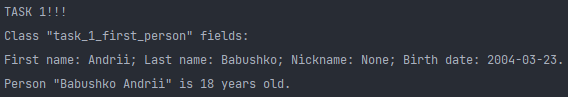
Примітка: при створенні атрибута birth\_date з рядка типу "2002-12-31" необхідно:

1. визначити яка інформація потрібна для створення об'єкта datetime.date,
2. отримати ці дані з рядка
3. розібрати її (дістати з неї окремо, рік, місяць, число),
4. на підставі цієї інформації створити спеціальний об'єкт datetime.date,
5. помістити цей спец. об'єкт в атрибут екземпляра класу

***Лістинг програми:***

*""" Lab 7. Python. Andrii Babushko. Repository: https://github.com/AndriiBabushko/Python """  
from* datetime *import* date  
  
  
*# task 1  
class* Person:  
 personCount: int = 0  
  
 *def \_\_init\_\_*(*self*, first\_name: str, last\_name: str, birth\_date: str, nickname=*None*):  
 *self*.firstName = first\_name  
 *self*.lastName = last\_name  
 *self*.nickname = nickname  
 *try*:  
 dates = birth\_date.split('-')  
 *self*.birthDate = date(int(dates[0]), int(dates[1]), int(dates[2]))  
 *except* ValueError *as* valueError:  
 print(f'Incorrect birth date! {valueError}')  
 *self*.birth\_date = *None* Person.personCount += 1  
  
 *def \_\_str\_\_*(*self*) -> str:  
 *return* f'First name: {*self*.firstName}; Last name: {*self*.lastName}; Nickname: {*self*.nickname}; Birth date: {*self*.birthDate}.'  
  
 *def* getAge(*self*) -> str:  
 *try*:  
 age = date.today() - *self*.birthDate  
 *return* f'{int(age.days / 365)}'  
 *except* ValueError *as* valueError:  
 *return* f'Incorrect birth date! {valueError}'  
  
 *def* getFullName(*self*) -> str:  
 *return* f'{*self*.lastName} {*self*.firstName}'  
  
  
*def* enterPersons(count: int) -> list[Person]:  
 counter = 0  
 personsList: list[Person] = []  
  
 *while* count > counter:  
 *try*:  
 firstName: str = input("Enter person's first name: ")  
 lastName: str = input("Enter person's last name: ")  
 birthDate: str = input("Enter person's birth date as 'YYYY-MM-DD': ")  
 nickname: str = input("Enter person's nickname(optional): ")  
 personsList.append(Person(firstName, lastName, birthDate, nickname))  
 count -= 1  
 *except* ValueError *as* valueError:  
 personsList.pop()  
 print(f'ERROR! {valueError}')  
  
 *return* personsList  
  
  
*def* enterCount(subject: str) -> int:  
 *while True*:  
 *try*:  
 count = int(input(f'Enter {subject}\'s count value: '))  
 *if* int(count):  
 *pass  
 return* count  
 *else*:  
 *raise* ValueError('ERROR! Something occurred!')  
 *except* ValueError *as* valueError:  
 print(f'ERROR! {valueError}')  
  
  
print('TASK 1!!!')  
*# task\_1\_persons\_list = enterPersons(enterCount('person'))  
# print(task\_1\_persons\_list[0])*task\_1\_first\_person: Person = Person('Andrii', 'Babushko', '2004-03-23')  
print(f'Class "task\_1\_first\_person" fields:\n{task\_1\_first\_person}')  
print(f'Person "{task\_1\_first\_person.getFullName()}" is {task\_1\_first\_person.getAge()} years old.')

***Результат програми:***

******

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

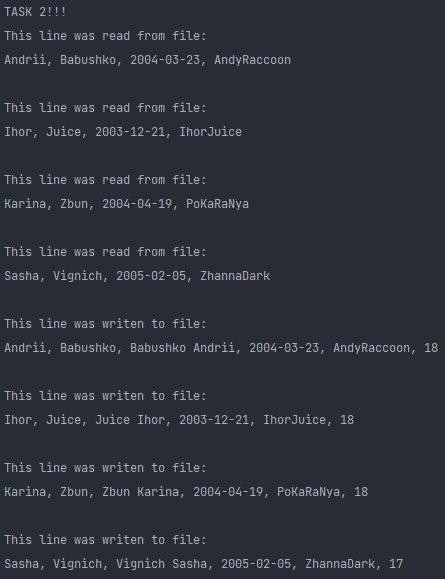
1. прочитати дані з переданого файлу;
2. створити об'єкти класу Person на підставі отриманих даних;
3. модифікувати дані в файлі
   1. додати графу повного імені (fullname) після графи з ім'ям (name)
   2. додати графу з віком (age) в кінець.

На виході отримати файл, розширений зазначеним чином.

***Лістинг програми:***

*# task 2  
def* modifier(fileName: str) -> list[Person]:  
 *import* io  
 *import* re  
  
 persons: list[Person] = []  
  
 *with* io.open(f'./{fileName}.txt', 'rt', encoding='utf-8') *as* personsData:  
 *for* person *in* personsData:  
 data: list[str] = re.split(', |\n', person)  
 print(f'This line was read from file:\n{person}')  
 persons.append(Person(data[0], data[1], data[2], data[3]))  
  
 *with* io.open(f'./new\_{fileName}.txt', 'wt', encoding='utf-8') *as* newPersonsData:  
 *for* person *in* persons:  
 written\_line: str = f'{person.firstName}, {person.lastName}, {person.getFullName()}, {person.birthDate}, {person.nickname}, {person.getAge()}\n'  
 newPersonsData.write(written\_line)  
 print(f'This line was writen to file:\n{written\_line}')  
  
 *return* persons  
  
  
print('\nTASK 2!!!')  
task\_2\_person\_list: list[Person] = modifier('persons\_data')

***Результат програми:***

******

***Увесь лістинг програми:***

*""" Lab 7. Python. Andrii Babushko. Repository: https://github.com/AndriiBabushko/Python """  
from* datetime *import* date  
  
  
*# task 1  
class* Person:  
 personCount: int = 0  
  
 *def \_\_init\_\_*(*self*, first\_name: str, last\_name: str, birth\_date: str, nickname=*None*):  
 *self*.firstName = first\_name  
 *self*.lastName = last\_name  
 *self*.nickname = nickname  
 *try*:  
 dates = birth\_date.split('-')  
 *self*.birthDate = date(int(dates[0]), int(dates[1]), int(dates[2]))  
 *except* ValueError *as* valueError:  
 print(f'Incorrect birth date! {valueError}')  
 *self*.birth\_date = *None* Person.personCount += 1  
  
 *def \_\_str\_\_*(*self*) -> str:  
 *return* f'First name: {*self*.firstName}; Last name: {*self*.lastName}; Nickname: {*self*.nickname}; Birth date: {*self*.birthDate}.'  
  
 *def* getAge(*self*) -> str:  
 *try*:  
 age = date.today() - *self*.birthDate  
 *return* f'{int(age.days / 365)}'  
 *except* ValueError *as* valueError:  
 *return* f'Incorrect birth date! {valueError}'  
  
 *def* getFullName(*self*) -> str:  
 *return* f'{*self*.lastName} {*self*.firstName}'  
  
  
*def* enterPersons(count: int) -> list[Person]:  
 counter = 0  
 personsList: list[Person] = []  
  
 *while* count > counter:  
 *try*:  
 firstName: str = input("Enter person's first name: ")  
 lastName: str = input("Enter person's last name: ")  
 birthDate: str = input("Enter person's birth date as 'YYYY-MM-DD': ")  
 nickname: str = input("Enter person's nickname(optional): ")  
 personsList.append(Person(firstName, lastName, birthDate, nickname))  
 count -= 1  
 *except* ValueError *as* valueError:  
 personsList.pop()  
 print(f'ERROR! {valueError}')  
  
 *return* personsList  
  
  
*def* enterCount(subject: str) -> int:  
 *while True*:  
 *try*:  
 count = int(input(f'Enter {subject}\'s count value: '))  
 *if* int(count):  
 *pass  
 return* count  
 *else*:  
 *raise* ValueError('ERROR! Something occurred!')  
 *except* ValueError *as* valueError:  
 print(f'ERROR! {valueError}')  
  
  
print('TASK 1!!!')  
*# task\_1\_persons\_list = enterPersons(enterCount('person'))  
# print(task\_1\_persons\_list[0])*task\_1\_first\_person: Person = Person('Andrii', 'Babushko', '2004-03-23')  
print(f'Class "task\_1\_first\_person" fields:\n{task\_1\_first\_person}')  
print(f'Person "{task\_1\_first\_person.getFullName()}" is {task\_1\_first\_person.getAge()} years old.')  
  
  
*# task 2  
def* modifier(fileName: str) -> list[Person]:  
 *import* io  
 *import* re  
  
 persons: list[Person] = []  
  
 *with* io.open(f'./{fileName}.txt', 'rt', encoding='utf-8') *as* personsData:  
 *for* person *in* personsData:  
 data: list[str] = re.split(', |\n', person)  
 print(f'This line was read from file:\n{person}')  
 persons.append(Person(data[0], data[1], data[2], data[3]))  
  
 *with* io.open(f'./new\_{fileName}.txt', 'wt', encoding='utf-8') *as* newPersonsData:  
 *for* person *in* persons:  
 written\_line: str = f'{person.firstName}, {person.lastName}, {person.getFullName()}, {person.birthDate}, {person.nickname}, {person.getAge()}\n'  
 newPersonsData.write(written\_line)  
 print(f'This line was writen to file:\n{written\_line}')  
  
 *return* persons  
  
  
print('\nTASK 2!!!')  
task\_2\_person\_list: list[Person] = modifier('persons\_data')

***Висновок:*** під час виконання лабораторної роботи було отримано навички створення свого класу, конструктора класу та методів всередині цього класу для роботи з його атрибутами.