Міністерство освіти і науки України
Державний університет „Житомирська політехніка”

Кафедра КІ та КБ
Група: КБ-21-1

Програмування мовою Python
Лабораторна робота №7
«КЛАСИ. Ч. 1»

Виконала: Поліщук К. Р.
Прийняв: Морозов Д. С.

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.125.10.000 –Лр.7		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Поліщук К.Р.			Звіт з лабораторної роботи №7	Літ.	Арк. 1 / Аркушів 5
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: 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'єкт в атрибут екземпляра класу

Лістинг програми:

```
import datetime
import codecs
from tabulate import tabulate
from class_person import ClassPerson

def task1():
    test = ClassPerson ('Поліщук', 'Карина',
                        datetime.datetime.strptime(str(input('Введіть дату [РР-ММ-ДД] ->
')), "%Y-%m-%d").date(),
                        'karina67')

    print(f"Вік контакту: {test.get_age()} років")
    print(f"Псевдонім контакту: {test.nickname}")
    print(test.get_fullname())
```

Результат виконання програми:

```
Введіть дату [РР-ММ-ДД] -> 2005-06-12
Вік контакту: 18 років
Псевдонім контакту: karina67
Поліщук Карина
```

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..

Лістинг програми:

```
def task2():
    with codecs.open('files\\persons.txt', 'w', "utf-8") as fileText:
            table = [['Поліщук', 'Карина', '2004-05-18'],
                     ['Кисіль', 'Остап', '2004-07-05'],
                     ['Догма', 'Тимур', '2003-09-29']]
            fileText.writelines(tabulate(table, headers=["Surname", "Name", "Birth
date"]))
            persons = ClassPerson.modifier('persons')
```

Результат виконання програми:

main.py | personsModifier.txt | class_person.py

```
1  Surname    Name       Fullname            Birth date      Age
2  ---------  ---------  ------------------  ------------  -----
3  Поліщук    Карина     Поліщук Карина      2004-05-18       18
4  Кисіль     Остап      Кисіль Остап        2004-07-05       18
5  Догма      Тимур      Догма Тимур         2003-09-29       19
```

persons.txt

```
1  Surname    Name    Birth date
2  ---------  ------  ------------
3  Поліщук    Карина  2004-05-18
4  Кисіль     Остап   2004-07-05
5  Догма      Тимур   2003-09-29
```

Лістинг класу Person:

```
from dateutil.relativedelta import relativedelta
import datetime
import codecs
from tabulate import tabulate

class ClassPerson:
    """
      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)
      """

    def __init__(self, surname, first_name, birth_date, nickname=''):
        self.surname = surname
        self.first_name = first_name
        self.nickname = nickname
        self.birth_date = birth_date

    def get_age(self):
       return (datetime.datetime.now() -
relativedelta(years=self.birth_date.year)).year
```

```
    def get_fullname(self):
        return self.surname + ' ' + self.first_name

    @staticmethod
    def modifier(filename):
        persons = []
        with codecs.open(f'files\\{filename}.txt', 'r', "utf-8") as fileRead:
            with codecs.open(f'files\\{filename}Modifier.txt', 'w', "utf-8") as
fileWrite:
                table = []
                count = 0
                for i in fileRead.readlines():
                    if count > 1:
                        line = i.split()
                        persons.append(
                            ClassPerson(line[0], line[1],
datetime.datetime.strptime(line[2], "%Y-%m-%d").date()))
                        table.append([line[0], line[1], persons[count -
2].get_fullname(), line[2],
                                      persons[count - 2].get_age()])
                    count += 1
                headers = ['Surname', 'Name', 'Fullname', 'Birth date', 'Age']
                fileWrite.writelines(tabulate(table, headers=headers))
        return persons
```

Висновок: у даній лабораторній роботі ми ознайомилися з ООП в мові Python.