**Лабораторна робота №3**

**Тема:** ООП.

**Мета:** Використовуючи теоретичне підґрунтя про ООП у мові Python переробити програму телефонного довідника студентів використовуючи принципи ООП для формування відомостей про студентів.

**Завдання до лабораторної роботи**

Переробити функціональність телефонного довідника студентів групи, що був розроблений у Лабораторній роботі №2 використовуючи принципи ООП:

1) розробити клас Студент групи з відповідними атрибутами;

2) розробити клас Список групи, має містити не словники, як виконано в лабораторній роботі №2, а об’єкти класу Студент групи; додавання нового запису, видаленні існуючого чи зміна даних має бути виконана через методи класу Список групи.

3) розробити клас для роботи з файлами для зчитування початкової інформації про список групи та збереження інформації по завершенню програми.

4) список студентів має містити не словники, як виконано в лабораторній роботі №2, а об’єкти класу Студент групи;

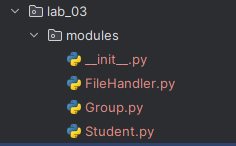
5) описання всіх класів мають міститися в окремих файлах, що мають відповідні імена(наприклад Studend, StudentList, Utils)

6) основний функціонал програми має бути покритий Юніт тестами.

**Хід роботи**

Ознайомившись з теоретичними відомостями та завданням розпочав виконувати лабораторну роботу.

Створив нові файли: FileHandler.py, Student.py, Group.py



Переписав довідник з використанням принципів ООП.

Файл FileHandler.py містить клас з методами завантаження та збереження даних студентів:

|  |
| --- |
| import csv from sys import argv from lab\_03.modules.Group import Group  class FileHandler:   def \_\_init\_\_(self, group:Group):  self.group = group  self.fileName = None   def loadArgs(self):  if (len(argv) == 2 and argv[1].endswith(".csv")):  self.fileName = argv[1]  else:  print("Можливий тільки 1 аргумент та він бути типа: file.csv")  exit()   def loadStudents(self):  with open(self.fileName) as file:  reader = csv.DictReader(file)  for row in reader:  self.group.addStudent(row["name"], row["surname"],row["phone"], row["email"])   print("Дані студентів завантажені!")   def saveStudents(self):  with open(self.fileName, "w", newline='') as file:  fieldnames = ["name", "surname", "phone", "email"]  writer = csv.DictWriter(file, fieldnames=fieldnames)  writer.writeheader()   for student in self.group.students:  writer.writerow({  "name": student.name,  "surname": student.surname,  "phone": student.phone,  "email": student.email  })  print("Збережено!") |

Файл Student.py містить клас з атрибутами студента:

|  |
| --- |
| class Student:   def \_\_init\_\_(self, name, surname, phone, email):  self.name = name  self.surname = surname  self.phone = phone  self.email = email   def \_\_str\_\_(self):  return f"Ім'я студента: '{self.name}', прізвище: {self.surname}, телефон: {self.phone}, пошта: {self.email})" |

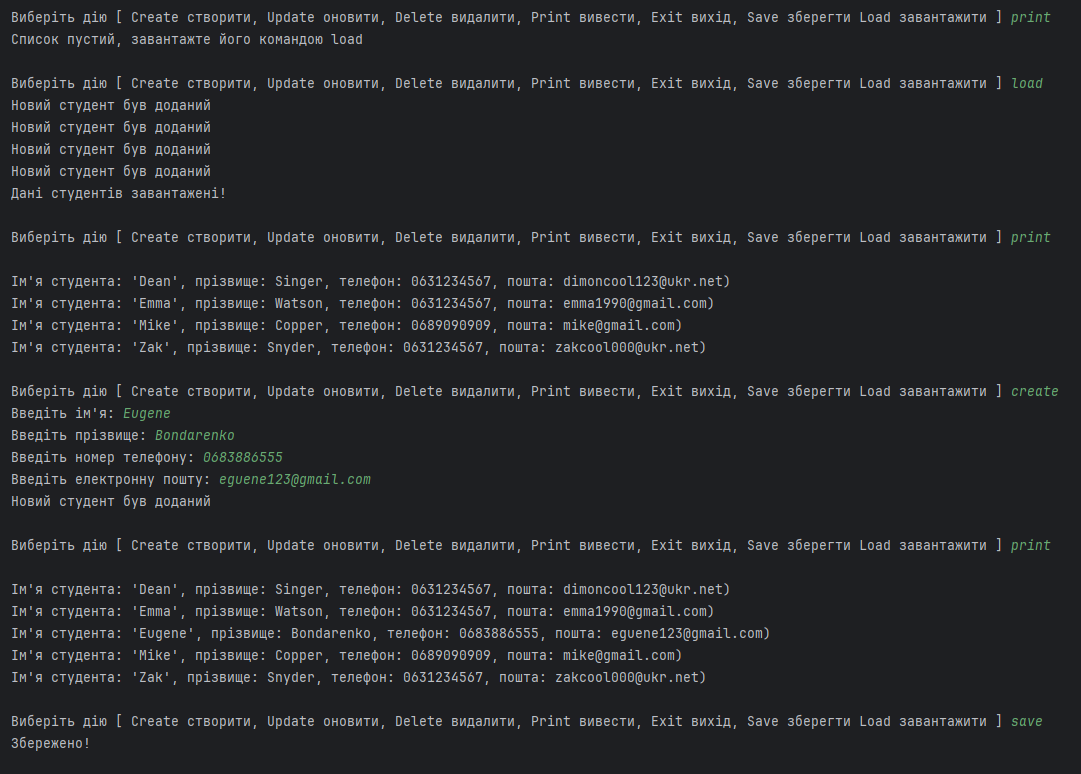
Файл Group.py містить клас з методами додавання, видалення, зміни даних студентів:

|  |
| --- |
| from lab\_03.modules.Student import Student  class Group:   def \_\_init\_\_(self):  self.students = []   def printStudents(self):  if (len(self.students) == 0):  print("Список пустий, завантажте його командою load")  else:  print(" ")  for elem in self.students:  print(elem)   def askParameters(self):  name = input("Введіть ім'я: ")  surname = input("Введіть прізвище: ")  phone = input("Введіть номер телефону: ")  email = input("Введіть електронну пошту: ")  return name, surname, phone, email   def addStudent(self, name, surname, phone, email):  student = Student(name, surname, phone, email)   insertPosition = 0  for item in self.students:  if name > item.name:  insertPosition += 1  else:  break   self.students.insert(insertPosition, student)  print("Новий студент був доданий")  return   def deleteStudent(self):  name = input("Введіть ім'я студента на видалення: ")  deletePosition = -1  for item in self.students:  if name == item.name:  deletePosition = self.students.index(item)  break  if deletePosition == -1:  print("Студента не знайдено")  else:  print("Видаляю позицію " + str(deletePosition))  del self.students[deletePosition]  print("Студент був видалений")  return   def updateStudent(self):  name = input("Введіть ім'я студента інформацію якого ви хочете змінити: ")   position = -1  for item in self.students:  if name == item.name:  position = self.students.index(item)  break  if position == -1:  print("Студента не знайдено")  else:   student = self.students[position]   name = input("Введіть нове ім'я. Натисніть ентер якщо не бажаєте це змінювати: ")  if not name:  name = student.name   surname = input("Введіть нове прізвище. Натисніть ентер якщо не бажаєте це змінювати: ")  if not surname:  surname = student.surname   phone = input("Введіть новий номер телефону. Натисніть ентер якщо не бажаєте це змінювати: ")  if not phone:  phone = student.phone   email = input("Введіть нову електронну пошту. Натисніть ентер якщо не бажаєте це змінювати: ")  if not email:  email = student.email   del self.students[position]   self.addStudent(name, surname, phone, email)   print("Інформація про студента оновлена") |

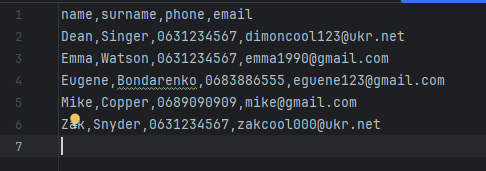
Файл запуску програми:

|  |
| --- |
| from lab\_03.modules.FileHandler import FileHandler from lab\_03.modules.Group import Group  def main():  group = Group()  fileHandler = FileHandler(group)  fileHandler.loadArgs()   while True:  choice = input("\nВиберіть дію [ Create створити, Update оновити, Delete видалити, Print вивести, Exit вихід, Save зберегти Load завантажити ] ")  match choice.lower().strip():  case "create":  group.addStudent(\*group.askParameters())  case "update":  group.updateStudent()  case "delete":  group.deleteStudent()  case "print":  group.printStudents()  case "exit":  break  case "save":  fileHandler.saveStudents()  case "load":  fileHandler.loadStudents()  case \_:  print("Неправильна команда!")  main() |

Результат роботи програми:



Збережений файл studends.csv:

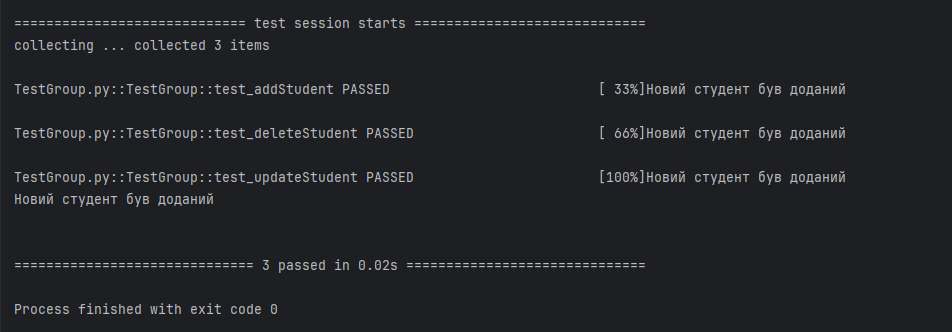


Наступним завданням було покрити основний функціонал юніт тестами.

Створив новий файл TestGroup.py та написав код для перевірки правильності виконання:

import unittest  
from unittest.mock import patch  
  
from lab\_03.modules.Group import Group  
  
class TestGroup(unittest.TestCase):  
  
 def setUp(self):  
 self.group = Group()  
  
 @patch('builtins.input', side\_effect=["Test", "Testovich", "123456789", "test@test.com"])  
 def test\_addStudent(self, mock\_input):  
 name, surname, phone, email = self.group.askParameters()  
  
 self.group.addStudent(name, surname, phone, email)  
  
 assert len(self.group.students) == 1  
 assert self.group.students[0].name == "Test"  
 assert self.group.students[0].surname == "Testovich"  
  
 @patch('builtins.input', side\_effect=["Test", "TestNewName", "NewSurname", "987654321", "new@new.com"])  
 def test\_updateStudent(self, mock\_input):  
 self.test\_addStudent()  
 self.group.updateStudent()  
  
 assert self.group.students[0].name == "TestNewName"  
 assert self.group.students[0].surname == "NewSurname"  
 assert self.group.students[0].phone == "987654321"  
 assert self.group.students[0].email == "new@new.com"  
  
  
 @patch('builtins.input', side\_effect=["Test"])  
 def test\_deleteStudent(self, mock\_input):  
 self.test\_addStudent()  
 self.group.deleteStudent()  
  
 assert len(self.group.students) == 0

Результат виконання був успішним, отже працює правильно:



**Висновок:** На цій лабораторній роботі я вдосконалив попередню версію своєї програми шляхом застовування принципів ООП та покрив функціонал свого коду юніт тестами.