# **Лабораторна робота №3**

ООП

**Мета роботи:** 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.

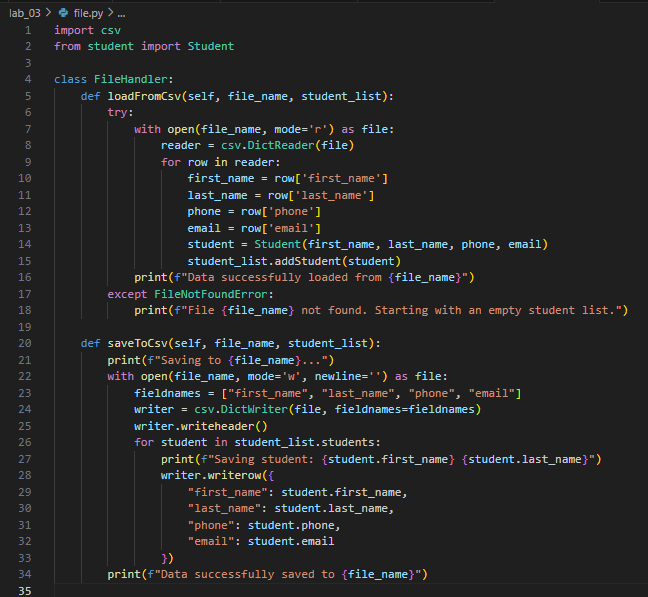
4) список студентів має містити не словники, як виконано в лабораторній роботі №2, а об’єкти класу Студент групи;

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

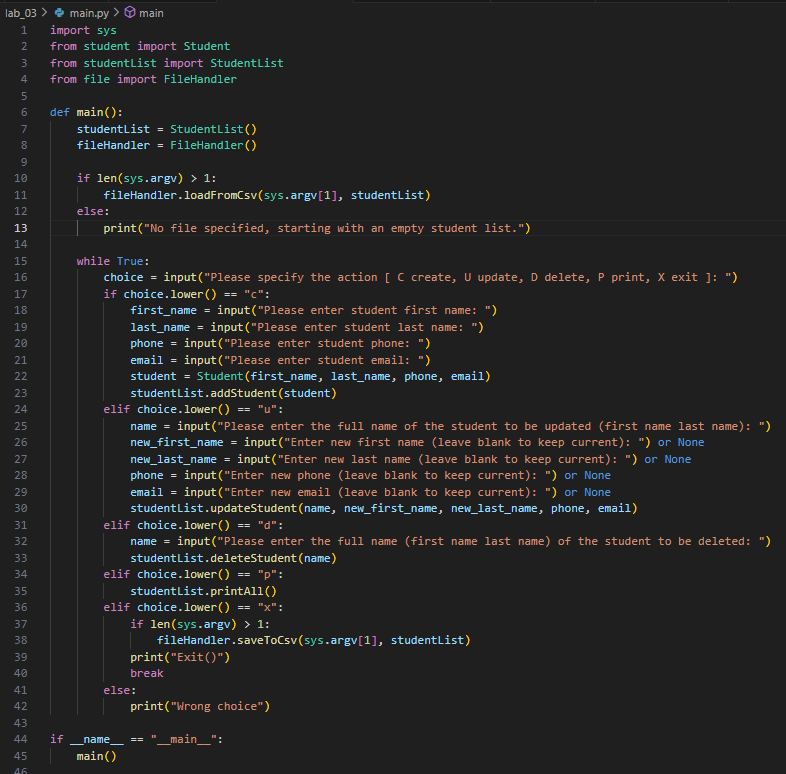
6) основний функціонал програми має бути покритий Юніт тестами.

Хід виконання завдання:

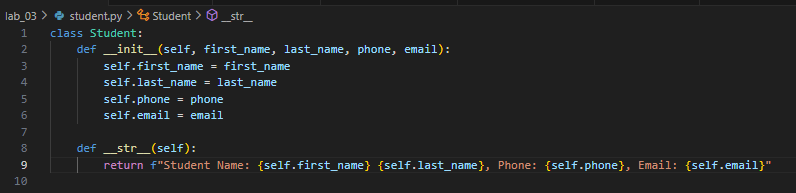
Файл file.py



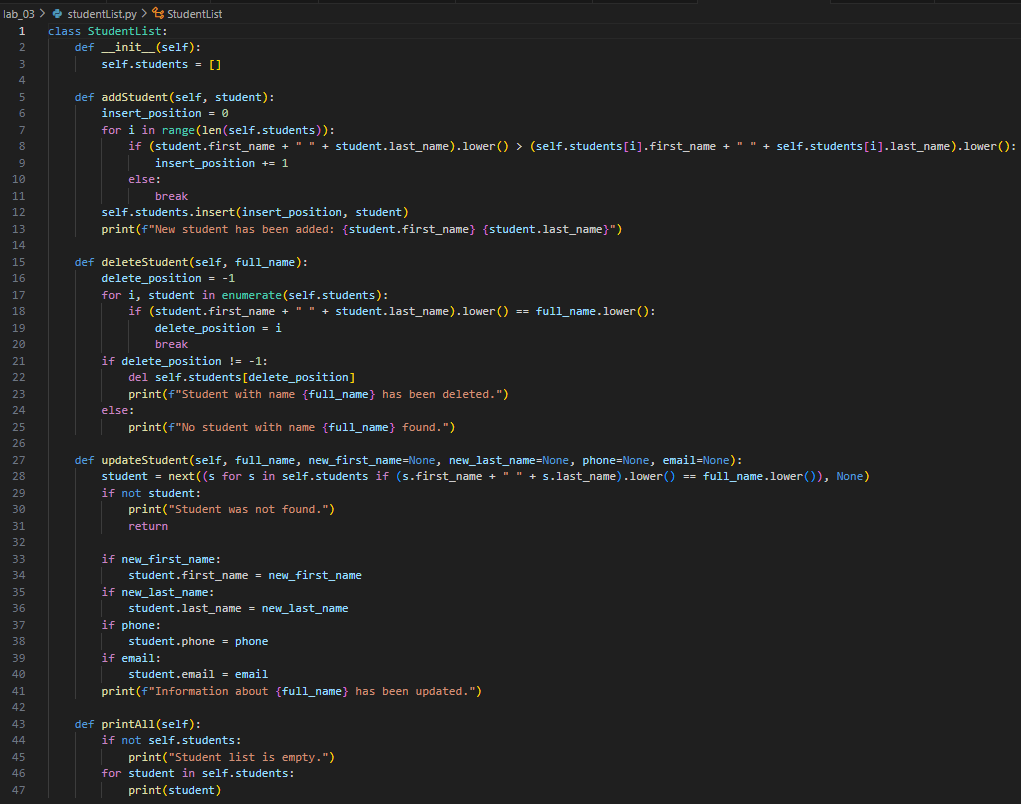
Файл main.py



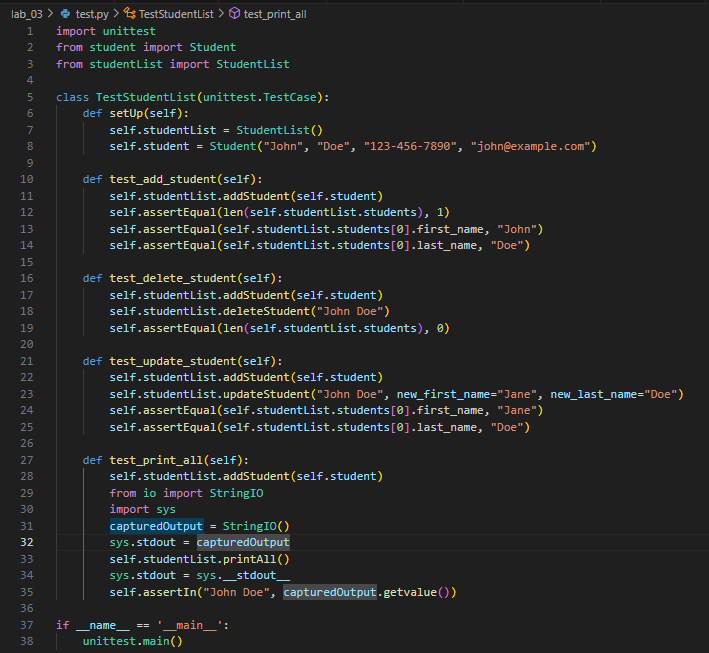
Файл student.py



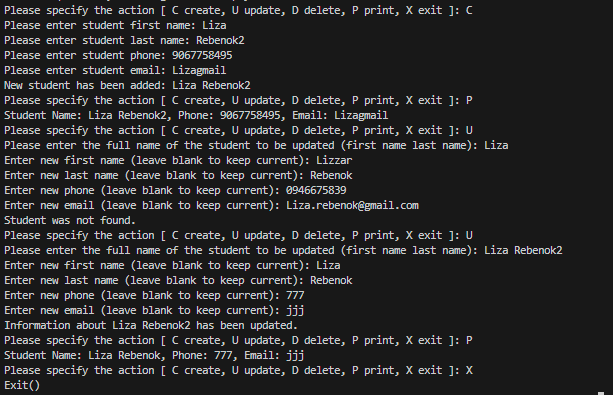
Файл studentList.py

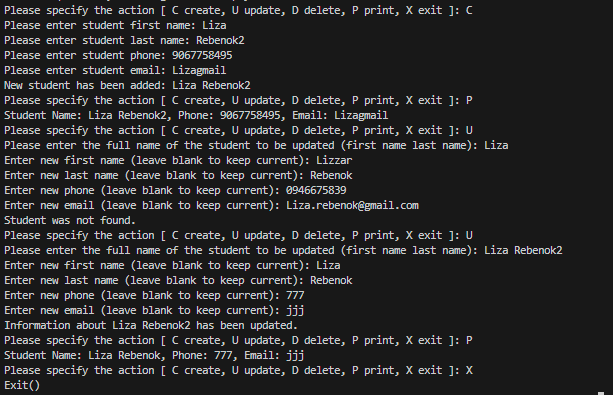


Файл test.py

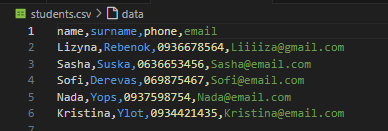


Вивід:





Кінцевий вигляд файлу students.csv



Кінцевий результат перевірки:



Текст Файлу file.py :

|  |
| --- |
| import csv  from student import Student  class FileHandler:  def loadFromCsv(self, file\_name, student\_list):  try:  with open(file\_name, mode='r') as file:  reader = csv.DictReader(file)  for row in reader:  first\_name = row['first\_name']  last\_name = row['last\_name']  phone = row['phone']  email = row['email']  student = Student(first\_name, last\_name, phone, email)  student\_list.addStudent(student)  print(f"Data successfully loaded from {file\_name}")  except FileNotFoundError:  print(f"File {file\_name} not found. Starting with an empty student list.")  def saveToCsv(self, file\_name, student\_list):  print(f"Saving to {file\_name}...")  with open(file\_name, mode='w', newline='') as file:  fieldnames = ["first\_name", "last\_name", "phone", "email"]  writer = csv.DictWriter(file, fieldnames=fieldnames)  writer.writeheader()  for student in student\_list.students:  print(f"Saving student: {student.first\_name} {student.last\_name}")  writer.writerow({  "first\_name": student.first\_name,  "last\_name": student.last\_name,  "phone": student.phone,  "email": student.email  })  print(f"Data successfully saved to {file\_name}") |

Текст Файлу main.py:

|  |
| --- |
| import sys  from student import Student  from studentList import StudentList  from file import FileHandler  def main():  studentList = StudentList()  fileHandler = FileHandler()  if len(sys.argv) > 1:  fileHandler.loadFromCsv(sys.argv[1], studentList)  else:  print("No file specified, starting with an empty student list.")    while True:  choice = input("Please specify the action [ C create, U update, D delete, P print, X exit ]: ")  if choice.lower() == "c":  first\_name = input("Please enter student first name: ")  last\_name = input("Please enter student last name: ")  phone = input("Please enter student phone: ")  email = input("Please enter student email: ")  student = Student(first\_name, last\_name, phone, email)  studentList.addStudent(student)  elif choice.lower() == "u":  name = input("Please enter the full name of the student to be updated (first name last name): ")  new\_first\_name = input("Enter new first name (leave blank to keep current): ") or None  new\_last\_name = input("Enter new last name (leave blank to keep current): ") or None  phone = input("Enter new phone (leave blank to keep current): ") or None  email = input("Enter new email (leave blank to keep current): ") or None  studentList.updateStudent(name, new\_first\_name, new\_last\_name, phone, email)  elif choice.lower() == "d":  name = input("Please enter the full name (first name last name) of the student to be deleted: ")  studentList.deleteStudent(name)  elif choice.lower() == "p":  studentList.printAll()  elif choice.lower() == "x":  if len(sys.argv) > 1:  fileHandler.saveToCsv(sys.argv[1], studentList)  print("Exit()")  break  else:  print("Wrong choice")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

Текст Файлу student.py:

|  |
| --- |
| class Student:  def \_\_init\_\_(self, first\_name, last\_name, phone, email):  self.first\_name = first\_name  self.last\_name = last\_name  self.phone = phone  self.email = email  def \_\_str\_\_(self):  return f"Student Name: {self.first\_name} {self.last\_name}, Phone: {self.phone}, Email: {self.email}" |

Текст Файлу studentList.py:

|  |
| --- |
| class StudentList:  def \_\_init\_\_(self):  self.students = []  def addStudent(self, student):  insert\_position = 0  for i in range(len(self.students)):  if (student.first\_name + " " + student.last\_name).lower() > (self.students[i].first\_name + " " + self.students[i].last\_name).lower():  insert\_position += 1  else:  break  self.students.insert(insert\_position, student)  print(f"New student has been added: {student.first\_name} {student.last\_name}")  def deleteStudent(self, full\_name):  delete\_position = -1  for i, student in enumerate(self.students):  if (student.first\_name + " " + student.last\_name).lower() == full\_name.lower():  delete\_position = i  break  if delete\_position != -1:  del self.students[delete\_position]  print(f"Student with name {full\_name} has been deleted.")  else:  print(f"No student with name {full\_name} found.")  def updateStudent(self, full\_name, new\_first\_name=None, new\_last\_name=None, phone=None, email=None):  student = next((s for s in self.students if (s.first\_name + " " + s.last\_name).lower() == full\_name.lower()), None)  if not student:  print("Student was not found.")  return  if new\_first\_name:  student.first\_name = new\_first\_name  if new\_last\_name:  student.last\_name = new\_last\_name  if phone:  student.phone = phone  if email:  student.email = email  print(f"Information about {full\_name} has been updated.")  def printAll(self):  if not self.students:  print("Student list is empty.")  for student in self.students:  print(student) |

Текст Файлу test.py:

|  |
| --- |
| import unittest  from student import Student  from studentList import StudentList  class TestStudentList(unittest.TestCase):  def setUp(self):  self.studentList = StudentList()  self.student = Student("John", "Doe", "123-456-7890", "john@example.com")  def test\_add\_student(self):  self.studentList.addStudent(self.student)  self.assertEqual(len(self.studentList.students), 1)  self.assertEqual(self.studentList.students[0].first\_name, "John")  self.assertEqual(self.studentList.students[0].last\_name, "Doe")  def test\_delete\_student(self):  self.studentList.addStudent(self.student)  self.studentList.deleteStudent("John Doe")  self.assertEqual(len(self.studentList.students), 0)  def test\_update\_student(self):  self.studentList.addStudent(self.student)  self.studentList.updateStudent("John Doe", new\_first\_name="Jane", new\_last\_name="Doe")  self.assertEqual(self.studentList.students[0].first\_name, "Jane")  self.assertEqual(self.studentList.students[0].last\_name, "Doe")  def test\_print\_all(self):  self.studentList.addStudent(self.student)  from io import StringIO  import sys  capturedOutput = StringIO()  sys.stdout = capturedOutput  self.studentList.printAll()  sys.stdout = sys.\_\_stdout\_\_  self.assertIn("John Doe", capturedOutput.getvalue())  if \_\_name\_\_ == '\_\_main\_\_':  unittest.main() |

Посилання на github:

<https://github.com/Lizarebenokk>