**Лабораторна робота №3**

**Тема**: ООП.

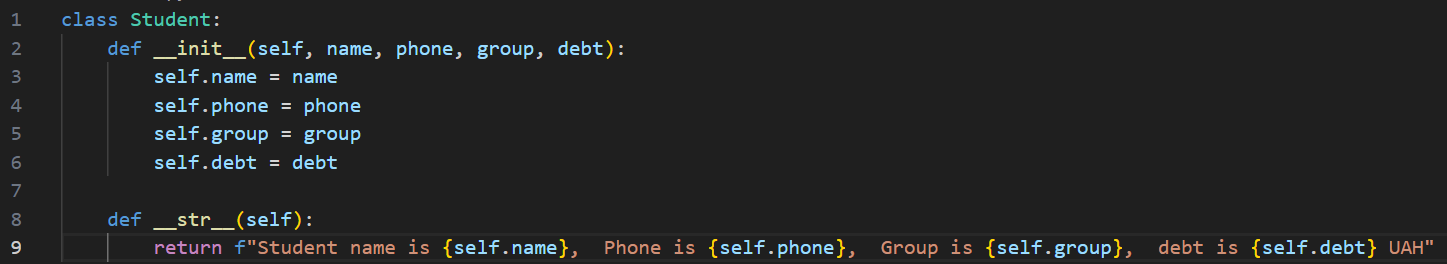
**Мета**: Використовуючи теоретичне підґрунтя про ООП у мові Python переробити програму телефонного довідника студентів використовуючи принципи ООП для формування відомостей про студентів.

**Завдання до лабораторної роботи**

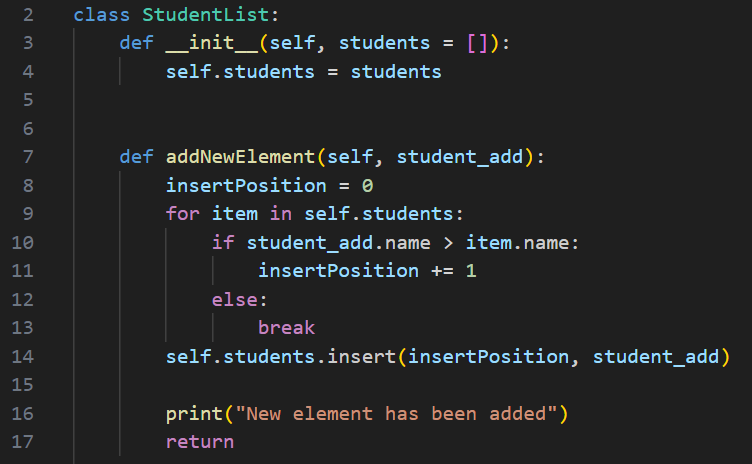
1. розробити клас Студент групи з відповідними атрибутами;
2. розробити клас Список групи, має містити не словники, як виконано в лабораторній роботі №2, а об’єкти класу Студент групи; додавання нового запису, видалення існуючого чи зміна даних має бути виконана через методи класу Список групи.
3. розробити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

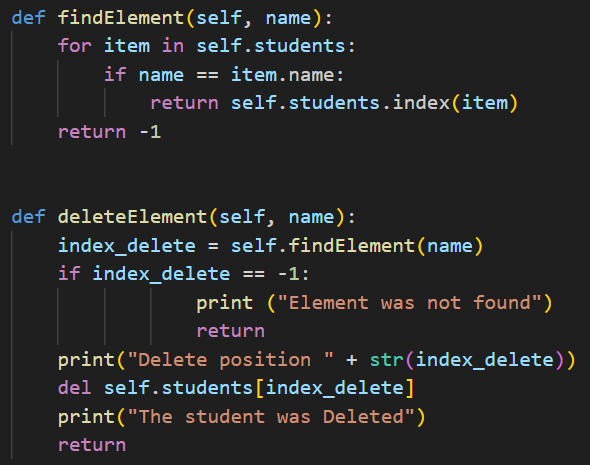
**Хід роботи**

Для початку я створив клас Student з відповідними атрибутами в окремому файлі, якщо роздрукувати об’єкт цього класу, то буде виведено значення його атрибутів:

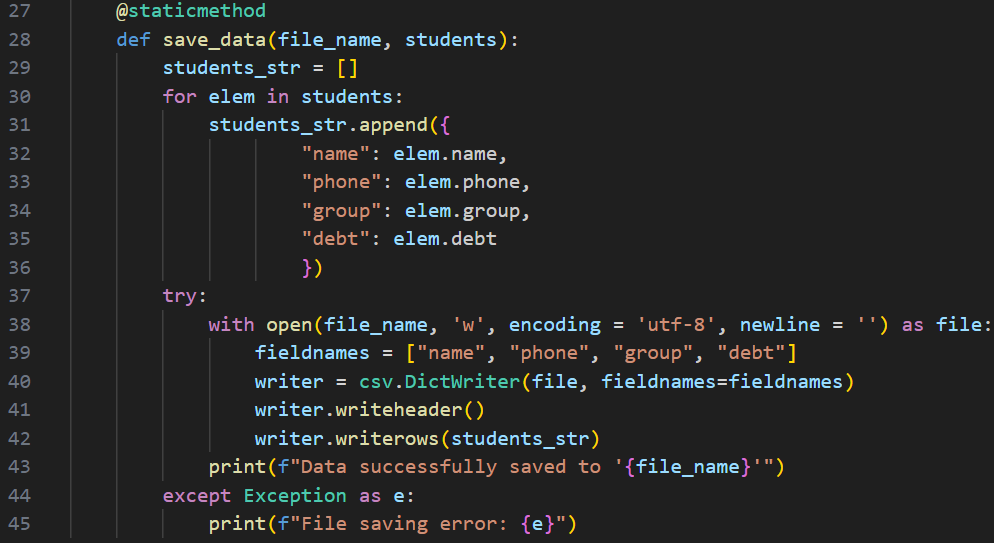


Далі створив клас StudentList для роботи зі списком студентів, в ньому містяться всі функції для роботи зі списком, додатково я трохи змінив структуру додавши метод пошуку елементу, тому що він буде потрібен для методів видалення та зміни елементів і також для виведення інформації в основному файлі перед зміною елемента, інші методи не зазнали змін з минулої роботи:

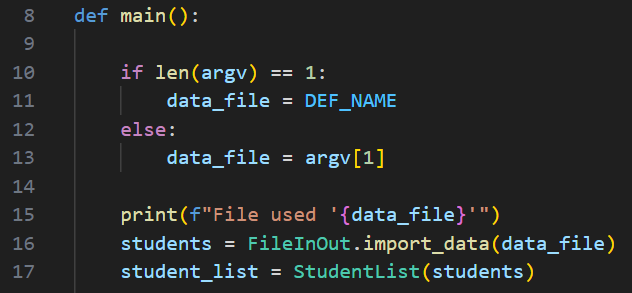




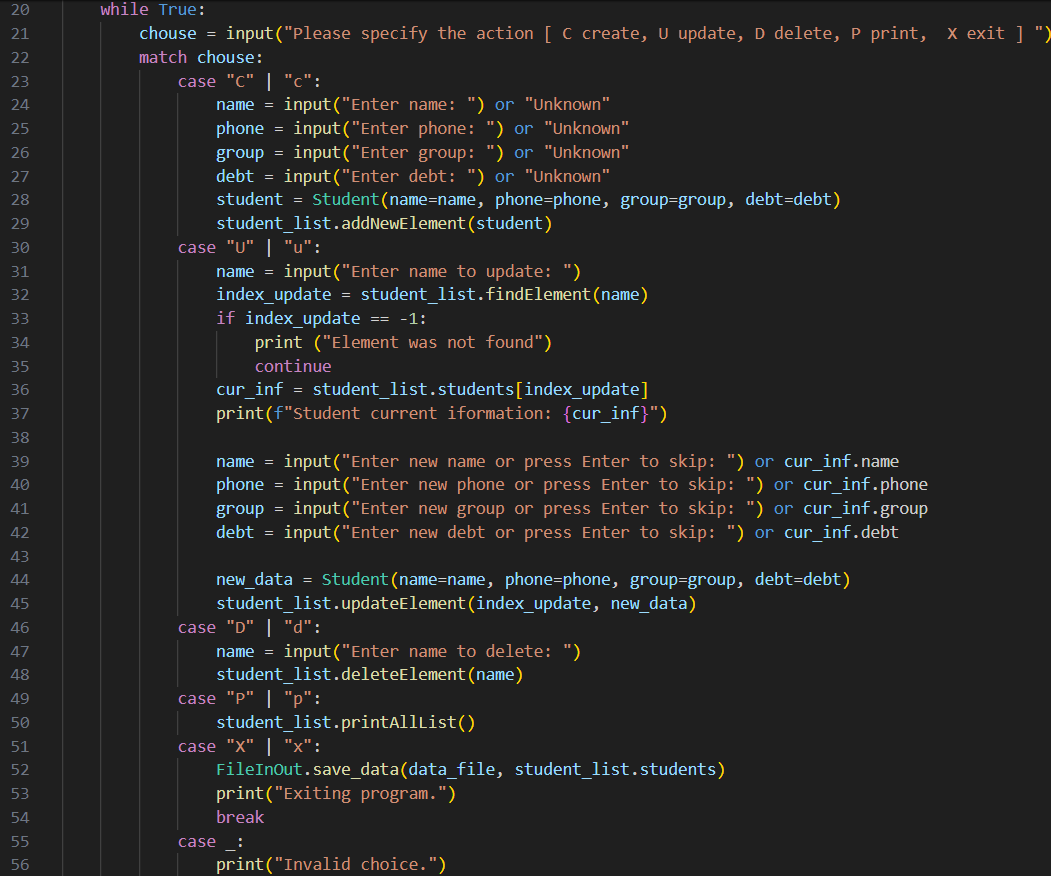
Далі йде клас FileInOut для роботи з файлом, він складається зі статичних методів, бо не потребує атрибутів, це просто 2 функції загорнуті в 1 клас, зміни зазанав лише метод збереження даних, він перезбирає список, щоб він містив словники, а не об’єкти класу, після цього цей новий список просто записується в файл:



Останнім файлом в цій програмі буде основний в ньому було трохи змінено структуру, тепер він записує початкові дані в змінну students, після чого створюється змінна student\_list, що э об’єктом класу StudentList, який приймає записані раніше початкові дані:



Тепер всі введення даних користувачем виконуються в цій функції, а потім вводяться в методи класів, тут можна побачити, що при оновленні даних спочатку шукається елемент для зміни та виводятся його атрибути, після чого відбувається введення нових даних, як і раніше, формується елемент класу Student та вводиться в функцію оновлення даних:



Текст програми (Student.py):

|  |
| --- |
| class Student:      def \_\_init\_\_(self, name, phone, group, debt):          self.name = name          self.phone = phone          self.group = group          self.debt = debt      def \_\_str\_\_(self):          return f"Student name is {self.name},  Phone is {self.phone},  Group is {self.group},  debt is {self.debt} UAH" |

Текст програми (Student\_list.py):

|  |
| --- |
| class StudentList:      def \_\_init\_\_(self, students = []):          self.students = students      def addNewElement(self, student\_add):          insertPosition = 0          for item in self.students:              if student\_add.name > item.name:                  insertPosition += 1              else:                  break          self.students.insert(insertPosition, student\_add)          print("New element has been added")          return      def findElement(self, name):          for item in self.students:              if name == item.name:                  return self.students.index(item)          return -1          def deleteElement(self, name):          index\_delete = self.findElement(name)          if index\_delete == -1:                      print ("Element was not found")                      return          print("Delete position " + str(index\_delete))          del self.students[index\_delete]          print("The student was Deleted")          return      def updateElement(self, index, new\_data):          element = self.students[index]          if element.name == new\_data.name and element.phone == new\_data.phone and element.group == new\_data.group and element.debt == new\_data.debt:              print("you haven`t updated student information")              return          elif element.name == new\_data.name:              element.phone = new\_data.phone              element.group = new\_data.group              element.debt = new\_data.debt          else:                del self.students[index]              insertPosition = 0              for item in self.students:                  if new\_data.name > item.name:                      insertPosition += 1                  else:                      break              self.students.insert(insertPosition, new\_data)              print("information has been updated")          return      def printAllList(self):          if not self.students:              print("The list of students is empty.")              return          for student in self.students:              print(student)          return |

Текст програми (File\_in\_out.py):

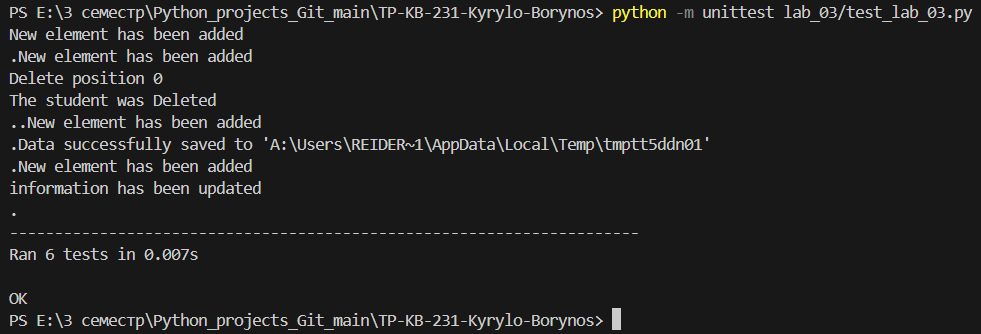
|  |
| --- |
| import csv  from .Student import Student  class FileInOut:      @staticmethod      def import\_data(file\_name):          students = []          try:              with open(file\_name, 'r', encoding = 'utf-8') as file:                  reader = csv.DictReader(file)                  for row in reader:                      student = Student(                          name = row.get("name", "Unknown"),                          phone = row.get("phone", "Unknown"),                          group = row.get("group", "Unknown"),                          debt = row.get("debt", "Unknown")                      )                      students.append(student)          except FileNotFoundError:              print(f"File '{file\_name}' not found \nThe initial list will be empty")          except Exception as e:              print(f"File loading error: {e}")          return students      @staticmethod      def save\_data(file\_name, students):          students\_str = []          for elem in students:              students\_str.append({                      "name": elem.name,                      "phone": elem.phone,                      "group": elem.group,                      "debt": elem.debt                      })          try:              with open(file\_name, 'w', encoding = 'utf-8', newline = '') as file:                  fieldnames = ["name", "phone", "group", "debt"]                  writer = csv.DictWriter(file, fieldnames=fieldnames)                  writer.writeheader()                  writer.writerows(students\_str)              print(f"Data successfully saved to '{file\_name}'")          except Exception as e:              print(f"File saving error: {e}") |

Текст програми (main.py):

|  |
| --- |
| from sys import argv  from Student import Student  from Student\_list import StudentList  from File\_in\_out import FileInOut  DEF\_NAME = "lab3\_data.csv"  def main():      if len(argv) == 1:          data\_file = DEF\_NAME      else:          data\_file = argv[1]        print(f"File used '{data\_file}'")      students = FileInOut.import\_data(data\_file)      student\_list = StudentList(students)      while True:          chouse = input("Please specify the action [ C create, U update, D delete, P print,  X exit ] ")          match chouse:              case "C" | "c":                  name = input("Enter name: ") or "Unknown"                  phone = input("Enter phone: ") or "Unknown"                  group = input("Enter group: ") or "Unknown"                  debt = input("Enter debt: ") or "Unknown"                  student = Student(name=name, phone=phone, group=group, debt=debt)                  student\_list.addNewElement(student)              case "U" | "u":                  name = input("Enter name to update: ")                  index\_update = student\_list.findElement(name)                  if index\_update == -1:                      print ("Element was not found")                      continue                  cur\_inf = student\_list.students[index\_update]                  print(f"Student current iformation: {cur\_inf}")                  name = input("Enter new name or press Enter to skip: ") or cur\_inf.name                  phone = input("Enter new phone or press Enter to skip: ") or cur\_inf.phone                  group = input("Enter new group or press Enter to skip: ") or cur\_inf.group                  debt = input("Enter new debt or press Enter to skip: ") or cur\_inf.debt                  new\_data = Student(name=name, phone=phone, group=group, debt=debt)                  student\_list.updateElement(index\_update, new\_data)              case "D" | "d":                  name = input("Enter name to delete: ")                  student\_list.deleteElement(name)              case "P" | "p":                  student\_list.printAllList()              case "X" | "x":                  FileInOut.save\_data(data\_file, student\_list.students)                  print("Exiting program.")                  break              case \_:                  print("Invalid choice.")  if \_\_name\_\_ == "\_\_main\_\_":      main() |

Останнім було покриття функціоналу юніт-тестами, тести такі ж самі, як і в 2 ЛР, але перероблені для роботи зі створенними класами:

Результат виконання тестів:



Текст програми:

|  |
| --- |
| import unittest  import unittest.mock  from io import StringIO  import sys  import os  import tempfile  import csv  from .Student import Student  from .Student\_list import StudentList  from .File\_in\_out import FileInOut  class TestStudenList(unittest.TestCase):      def setUp(self):          self.test\_file = tempfile.NamedTemporaryFile('w', encoding='utf-8',  newline='', delete=False)          self.test\_file\_name = self.test\_file.name          self.test\_file.close()          self.students\_list = StudentList()          self.students\_list.students = []      def tearDown(self):          if os.path.exists(self.test\_file\_name):              os.remove(self.test\_file\_name)      def test\_addNewElement(self):          input\_data = Student(name="Test", phone="0662835915", group="KB-231", debt="1500")          self.students\_list.addNewElement(input\_data)          self.assertEqual(len(self.students\_list.students), 1)          self.assertEqual(self.students\_list.students[0].name, "Test")          self.assertEqual(self.students\_list.students[0].phone, "0662835915")          self.assertEqual(self.students\_list.students[0].group, "KB-231")          self.assertEqual(self.students\_list.students[0].debt, "1500")      def test\_deleteElement(self):          input\_data = Student(name="Test", phone="0662835915", group="KB-231", debt="1500")          self.students\_list.addNewElement(input\_data)          self.students\_list.deleteElement("Test")          self.assertEqual(len(self.students\_list.students), 0)      def test\_updateElement(self):          input\_data = Student(name="Test", phone="0662835915", group="KB-231", debt="1500")          self.students\_list.addNewElement(input\_data)          updated\_data = Student(name="Test Updated", phone="0669999999", group="KB-232", debt="800")          self.students\_list.updateElement(0, updated\_data)          self.assertEqual(len(self.students\_list.students), 1)          self.assertEqual(self.students\_list.students[0].name, "Test Updated")          self.assertEqual(self.students\_list.students[0].phone, "0669999999")          self.assertEqual(self.students\_list.students[0].group, "KB-232")          self.assertEqual(self.students\_list.students[0].debt, "800")        def test\_import\_data(self):          data = [              {"name": "Alice", "phone": "0664682546", "group": "KB-231", "debt": "300"},              {"name": "Bob", "phone": "0666483595", "group": "KB-232", "debt": "700"}          ]          with open(self.test\_file\_name, 'w', encoding='utf-8', newline='') as file:              writer = csv.DictWriter(file, fieldnames=["name", "phone", "group", "debt"])              writer.writeheader()              writer.writerows(data)          students = FileInOut.import\_data(self.test\_file\_name)          self.assertEqual(len(students), 2)          self.assertEqual(students[0].name, "Alice")          self.assertEqual(students[1].name, "Bob")          self.assertEqual(students[0].group, "KB-231")          self.assertEqual(students[1].group, "KB-232")      def test\_save\_data(self):          student\_list = [ Student(name="Chris", phone="0667777777", group="KB-232", debt="500") ]          FileInOut.save\_data(self.test\_file\_name, student\_list)          with open(self.test\_file\_name, 'r', encoding='utf-8') as file:              reader = csv.DictReader(file)              rows = list(reader)              self.assertEqual(len(rows), 1)              self.assertEqual(rows[0]["name"], "Chris")              self.assertEqual(rows[0]["phone"], "0667777777")              self.assertEqual(rows[0]['group'], "KB-232")              self.assertEqual(rows[0]['debt'], "500")      def test\_printAllList(self):          input\_data = Student(name="Test", phone="0667777777", group="KB-232", debt="100")          self.students\_list.addNewElement(input\_data)          captured\_output = StringIO()          sys.stdout = captured\_output          self.students\_list.printAllList()          sys.stdout = sys.\_\_stdout\_\_          self.assertIn("Test", captured\_output.getvalue())  if \_\_name\_\_ == '\_\_main\_\_':      unittest.main() |

**Висновок:** під час виконання роботи я використовуючи теоретичне підґрунтя про ООП у мові Python переробив програму телефонного довідника студентів використовуючи принципи ООП для формування відомостей про студентів.