Технології програмування

Лабораторна робота №1

Тищенко Дмитро КБ-241

**Списки. Словники. Кортежі**

**Мета роботи**: Використовуючи теоретичне підґрунтя про складні структури даних Списки, Словники Кортежі, та використовуючи існуючий код, доробити програму додавши функціонал, що буде вказано в завданні до лабораторної роботи.

**Завдання до лабораторної роботи**

Реалізувати **відсортований** телефонний довідник студентів групи.

Для виконання задання надано частину готового функціоналу, яка розміщена в одній директорії з завданням до лабораторної роботи та має назву **lab\_01.py.**

Частина готового функціоналу реалізує безкінечний цикл запитів до користувача. Типи запитів: додати нового студента, змінити данні про існуючого студента, видалити запис, роздрукувати всю таблицю та вихід із програми. Реалізован функціонал додавання нового запису та видалення існуючого. Всі дії відбуваються з **відсортованим** списком студентів.

Перед виконанням роботи слід ознайомитись з існуючим функціоналом.

Необхідно розширити відомості про студента до 4х полів. На даний час використовується лише два поля (name та phone).

Необхідно переробити існуючий функціонал враховуючи розширення відомості про студента до 4х полів.

Необхідно реалізувати з нуля функціонал зміни інформації про студента враховуючи той факт, що вже існує реалізація додавання нового запису та видалення існуючого. **При зміні інформації про студента список має залишатись відсортованим**.

Текст програми разом зі звітом розмістити в каталозі lab\_01. Каталог lab\_01 розмістити в каталозі, що використовується для виконання практичних завдань по кожній лекції та має назву **TP-KB-22[1 or 2]-Name-Surname**.

Зроблений повний код програми:

|  |
| --- |
| # already sorted list  students = [  {"name": "Bob", "phone": "0631234567", "email": "bob@example.com", "group": "IP-21"},  {"name": "Emma", "phone": "0631234567", "email": "emma@example.com", "group": "IP-22"},  {"name": "Jon", "phone": "0631234567", "email": "jon@example.com", "group": "IP-21"},  {"name": "Zak", "phone": "0631234567", "email": "zak@example.com", "group": "IP-23"}  ]  def printAllList():  for elem in students:  strForPrint = (f"Name: {elem['name']}, Phone: {elem['phone']}, "  f"Email: {elem['email']}, Group: {elem['group']}")  print(strForPrint)  return  def addNewElement():  name = input("Please enter student name: ")  phone = input("Please enter student phone: ")  email = input("Please enter student email: ")  group = input("Please enter student group: ")  newItem = {"name": name, "phone": phone, "email": email, "group": group}  # find insert position  insertPosition = 0  for item in students:  if name > item["name"]:  insertPosition += 1  else:  break  students.insert(insertPosition, newItem)  print("New element has been added")  return  def deleteElement():  name = input("Please enter name to be deleted: ")  deletePosition = -1  for item in students:  if name == item["name"]:  deletePosition = students.index(item)  break  if deletePosition == -1:  print("Element was not found")  else:  del students[deletePosition]  print(f"Student '{name}' has been deleted")  return  def updateElement():  name = input("Please enter name to be updated: ")  updateIndex = -1  for item in students:  if name == item["name"]:  updateIndex = students.index(item)  break  if updateIndex == -1:  print("Element was not found")  return    print("Leave field empty if you don't want to change it.")  new\_name = input(f"New name [{students[updateIndex]['name']}]: ") or students[updateIndex]['name']  new\_phone = input(f"New phone [{students[updateIndex]['phone']}]: ") or students[updateIndex]['phone']  new\_email = input(f"New email [{students[updateIndex]['email']}]: ") or students[updateIndex]['email']  new\_group = input(f"New group [{students[updateIndex]['group']}]: ") or students[updateIndex]['group']  # Remove old record  del students[updateIndex]  # Create updated record  updatedItem = {  "name": new\_name,  "phone": new\_phone,  "email": new\_email,  "group": new\_group  }  # Re-insert into correct position to keep list sorted  insertPosition = 0  for item in students:  if new\_name > item["name"]:  insertPosition += 1  else:  break  students.insert(insertPosition, updatedItem)  print("Element has been updated")  return  def main():  while True:  chouse = input("Please specify the action [ C create, U update, D delete, P print, X exit ] ")  match chouse:  case "C" | "c":  print("New element will be created:")  addNewElement()  printAllList()  case "U" | "u":  print("Existing element will be updated")  updateElement()  printAllList()  case "D" | "d":  print("Element will be deleted")  deleteElement()  printAllList()  case "P" | "p":  print("List will be printed")  printAllList()  case "X" | "x":  print("Exit()")  break  case \_:  print("Wrong choice")  main() |

Фото реалізації видаленню елементу зі списку:

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Фото реалізації створення елементу у списку:

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Фото реалізації оновлення елементу у списку:

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Фото реалізації виводу усіх елементів списку:

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.