Технології програмування

Лабораторна робота №1

**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.

**Теоретичні відомості**

**Списки**

Масив – набір фіксованої кількості елементів, що розміщені в пам’яті комп’ютера безпосередньо один за одним, а доступ до них здійснюється за індексом (номер даного елементу в масиві).

В Python для реалізації масиву використовуються списки. Список – тип даних, що представляє собою послідовність певних значень, що можуть повторюватись. Але на відміну від масиву – кількість елементів у списку може бути довільною.

Списки – структура даних, що може містити елементи різних типів, що перераховані через кому та заключені в квадратні дужки.

Списки служать для того, щоб зберігати об’єкти в певному порядку, особливо якщо порядок або вміст можуть змінюватися. Можна змінювати список, додати в нього нові елементи, а також видалити або перезаписати існуючі. Можна змінити кількість елементів у списку, а також самі елементи. Одне і те ж значення може зустрічатися в списку кілька разів.

Приклад визначення списку

|  |
| --- |
| list\_num = ["1", "2", "3"]  print(list\_num)  list\_str = ["aa", "bb", "cc"]  print(list\_str) |

Результат

|  |
| --- |
| ['1', '2', '3']  ['aa', 'bb', 'cc'] |

Крім того, за допомогою функції list() можна створити порожній список

Список містить різні дані, звертатися до яких можна через ім’я списку та вказавши зміщення необхідного елементу

|  |
| --- |
| students = ["Ihor", "Dima", "Serge"]  print(students)  print(students[0]) |

Результат

|  |
| --- |
| ['Ihor', 'Dima', 'Serge']  Ihor |

Використовуючи методи списку можна виконувати необхідні операції. Для додавання елементів в кінець списку – використовують метод **append()**. Можна об’єднати один список з іншим за допомогою методу **extend()**. Функція **append()** додає елементи тільки в кінець списку. Коли потрібно додати елемент в задану позицію, використовується функція **insert().**

За допомогою функції **pop()** можна отримати елемент зі списку і в той же час видалити його. Якщо викликати функцію **pop()** і вказати зсув, вона поверне елемент, що знаходиться в заданій позиції. Якщо аргумент не вказано – буде використано значення -1. Так, виклик pop(0) поверне головний (початковий) елемент списку, а виклик pop() або pop(-1) – кінцевий елемент.

Для проходження по списку використовуються цикли

|  |
| --- |
| students = ["Ihor", "Dima", "Serge"]  for name in students:  print(name) |

Результат

|  |
| --- |
| Ihor  Dima  Serge |

**Словники**

Словник дуже схожий на список, але порядок елементів в ньому не має значення, і вони вибираються не за допомогою зміщення. Замість цього для кожного значення вказується пов’язаний з ним унікальний ключ. Таким ключем може бути об’єкт одного з незмінних типів: рядок, булева змінна, ціле число, число з плаваючою точкою, кортеж і іншими об’єктами. Елементи словника можуть містити об’єкти довільного типу даних і мати необмежений рівень вкладеності. Елементи в словниках розташовуються в довільному порядку.

Словники можна змінювати – це означає, що можна додати, видалити і змінити їх елементи, які мають вигляд "ключ – значення"

Щоб створити словник, необхідно заключити в фігурні дужки ({}) розділені комами пари ключ: значення.

|  |
| --- |
| animals = {  "dog" : 4,  "cat" : 4,  "goose": 2  }  print(animals) |

Результат

|  |
| --- |
| {'dog': 4, 'cat': 4, 'goose': 2} |

Можна використовувати функцію **dict**(), щоб створити порожній словник, якщо не вказати параметри функції

Звернення до елементів словника здійснюється за допомогою квадратних дужок, в яких вказується ключ.

|  |
| --- |
| animals = {  "dog" : 4,  "cat" : 4,  "goose": 2  }  print(animals["dog"]) |

Результат

|  |
| --- |
| 4 |

Щоб дізнатися, чи міститься в словнику якийсь ключ, використовується ключове слово **in**. Якщо ключ знайдений, то повертається значення True, в іншому випадку – **False**.

|  |
| --- |
| animals = {  "dog" : 4,  "cat" : 4,  "goose": 2  }  print("cat" in animals) |

Оскільки словники відносяться до змінюваних типів даних, то можна додати або змінити елемент по ключу. Додати елемент в словник досить легко. Потрібно просто звернутися до елементу по його ключу і привласнити йому значення. Якщо ключ вже існує в словнику, наявне значення буде замінено новим. Якщо ключ новий, він і вказане значення будуть додані в словник.

Для словників розроблено набір методів. **update()** – додає елементи в словник. Метод змінює поточний словник і нічого не повертає.

Видалити елемент зі словника можна за допомогою інструкції **del**.

|  |
| --- |
| dict\_2 = {"a": 1, "b": 2}  print(dict\_2)  del dict\_2 ["b"] # Видаляємо елемент з ключем "b"  print(dict\_2) |

Результат:

|  |
| --- |
| {'a': 1, 'b': 2}  {'a': 1} |

Щоб видалити всі ключі і значення зі словника, слід використовувати функцію **clear()** або просто привласнити порожній словник заданому імені.

Cкориставшись функцією **keys()** можна отримати всі ключі словника. Щоб отримати всі значення словника, використовується функція **values().** Щоб отримати всі пари "ключ – значення" із словника, використовується функція **items**().

|  |
| --- |
| testDisct = {"a": 1, "b": 2}  print(testDisct.keys())  print(testDisct.values())  print(testDisct.items()) |

Результат:

|  |
| --- |
| dict\_keys(['a', 'b'])  dict\_values([1, 2])  dict\_items([('a', 1), ('b', 2)]) |

**Кортежі**

Кортежі, як і списки, є послідовностями довільних елементів. На відміну від списків кортежі незмінні.

Всі операції над списками, що не змінюють список (додавання, множення на число, функції index() і count() і деякі інші операції) можна застосовувати до кортежів. Можна також по-різному змінювати елементи місцями і так далі.

Щоб створити порожній кортеж використовується оператор ().

|  |
| --- |
| xy = (12, 21)  print(xy) |

Результат

|  |
| --- |
| (12, 21) |

Функція перетворення tuple() створює кортежі з інших об’єктів

|  |
| --- |
| students = ['Alex', 'Helen', 'Olga']  print(students)  tuple\_students = tuple(students)  print(tuple\_students) |

Результат

|  |
| --- |
| ['Alex', 'Helen', 'Olga']  ('Alex', 'Helen', 'Olga') |

**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**.

**Хід виконання завдання:**

**Ознайомлення з теоретичним матеріалом**

* Вивчив основи роботи зі списками, словниками та кортежами.
* Розглянув надані приклади та методи роботи зі структурами даних.

**Аналіз початкового коду**

* Ознайомився з вихідним кодом файлу lab\_01.py.
* Виявив, що реалізовано функціонал додавання (addNewElement()) та видалення (deleteElement()) студентів зі списку.
* Визначив, що список list зберігає словники з полями name та phone, але потрібно розширити його до чотирьох полів (name, phone, email, address).
* Переконався, що всі операції зі списком підтримують його сортування за іменами студентів.

**Розширення структури даних**

* Оновив початковий список, додавши для кожного студента два нових поля: email та address.

**Модифікація функції додавання нового запису (addNewElement())**

* Додав запити на введення email та address.
* Забезпечив сортування списку після додавання нового елемента, щоб зберегти порядок студентів за іменем.

**Розробка функції оновлення даних (updateElement())**

* Додав можливість змінювати телефон, email та адресу студента, залишаючи інші дані незмінними, якщо користувач не вводить нові значення.
* Після оновлення даних сортував список, щоб зберегти його впорядкованість.

**Зміни у функції видалення (deleteElement())**

* Додав перевірку наявності студента перед видаленням.
* Забезпечив коректне видалення запису.

**Зміни у функції виведення списку (printAllList())**

* Додав виведення нових полів email та address.

**Доопрацювання головного циклу (main())**

* Додав команду U для оновлення даних студента.
* Забезпечив зручний інтерфейс користувача для вибору дій (C, U, D, P, X).
* Виправив незначні помилки (наприклад, неправильний варіант слова "choice").

Кінцевий варіант коду повністю відповідає вимогам лабораторної роботи: працює з розширеним телефонним довідником, підтримує всі необхідні операції, зберігає дані у відсортованому вигляді.

Текст програми:

|  |
| --- |
| list = [  {"name": "Bob", "phone": "0631234567", "email": "[bob@mail.com](mailto:bob@mail.com)", "address": "123 Elm St"},  {"name": "Emma", "phone": "0631234567", "email": "[emma@mail.com](mailto:emma@mail.com)", "address": "456 Oak St"},  {"name": "Jon", "phone": "0631234567", "email": "[jon@mail.com](mailto:jon@mail.com)", "address": "789 Pine St"},  {"name": "Zak", "phone": "0631234567", "email": "[zak@mail.com](mailto:zak@mail.com)", "address": "101 Maple St"}  ]  def printAllList():  for elem in list:  strForPrint = f"Student name is {elem['name']}, Phone is {elem['phone']}, Email is {elem['email']}, Address is {elem['address']}"  print(strForPrint)  return  def addNewElement():  name = input("Pease enter student name: ")  phone = input("Please enter student phone: ")  email = input("Please enter student email: ")  address = input("Please enter student address: ")  newItem = {"name": name, "phone": phone, "email": email, "address": address}  insertPosition = 0  for item in list:  if name > item["name"]:  insertPosition += 1  else:  break  list.insert(insertPosition, newItem)  print("New element has been added")  return  def deleteElement():  name = input("Please enter name to be delated: ")  deletePosition = -1  for item in list:  if name == item["name"]:  deletePosition = list.index(item)  break  if deletePosition == -1:  print("Element was not found")  else:  del list[deletePosition]  print("Element has been deleted")  return  def updateElement():  name = input("Please enter name to be updated: ")  for item in list:  if name == item["name"]:  print("Updating information. Leave fields blank to keep current values.")  new\_phone = input(f"Current phone ({item['phone']}): ")  new\_email = input(f"Current email ({item['email']}): ")  new\_address = input(f"Current address ({item['address']}): ")  item["phone"] = new\_phone if new\_phone else item["phone"]  item["email"] = new\_email if new\_email else item["email"]  item["address"] = new\_address if new\_address else item["address"]  list.sort(key=lambda x: x["name"])  print("Element has been updated")  return  print("Element not found")  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 chouse")  main() |

Посилання на github:

<https://github.com/worhels/lab/tree/main/01>

Знімок екрану з посилання на github:

