МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Програмування

Лабораторна робота №4

«Списки, кортежі, множини і діапазони в Python»

Виконав:
студент групи ІО-32
Крадожон М. Р.
Номер у списку групи: 16
Перевірив:
Пономаренко А. М.

Київ – 2023

Лабораторна робота №4

Тема: «Списки, кортежі, множини і діапазони в Python».

Мета: вивчити способи створення списків, кортежів, множин та задавання діапазонів. Операції над списками, кортежами та діапазонами. Функції для перетворень списків, кортежів та множин.

Загальне завдання:

1. Вивчити матеріал лекцій 11, 12, 13 та 14.
2. Виконати індивідуальне завдання лабораторної роботи, вибране відповідно до варіанту.

Короткі теоретичні основи:

1. Списки є змінюваними типами даних.
2. Кортежі є незмінюваними типами даних.
3. Діапазони є наборами чисел, сформованими на основі заданих початкового, кінцевого значень і величини кроку між числами.

Створити список можна такими способами:

1. Перелічивши всі елементи списку всередині квадратних дужок.
2. За допомогою генератора списків.

Операції над списками

1. Оператори + та +=
2. Оператори in та not in

Багатовимірні списки

Будь-який елемент списку може містити список.

Створити двовимірний список можна, наприклад, так:

```
>>> n = [[1,2,3], [4,5,6],[7,8,9]]
>>> n
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>>n[0][0]
1
>>>n[2][1]
8
```

Перебір елементів списку

1. За допомогою циклу for: `for i in arr: print(i, end=" ")`
2. За допомогою генераторів списків

```
>>> arr = [i for i in range(1,10)]
>>> arr
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Функція zip()

Вбудована функція zip() на кожній ітерації повертає кортеж, що містить елементи послідовностей, які розташовані на однаковому зсуві.

```
zip (<Послідовність1>[, ... , < ПослідовністьN>])
```

Функція sorted(<Послідовність>[, key=None] [, reverse=False]) – формує новий відсортований список, а початковий залишає без змін.

Функція max() визначає максимальне значення кортежу.

Створити кортеж можна указавши всі елементи через кому всередині круглих дужок.

Для створення діапазону застосовується функція range():

```
range ([<Початок>,] <Кінець> [, <Крок>])
```

Завдання 1:

Відповідно до номера в списку вибрати індивідуальне завдання. Написати програму мовою Python. Забезпечити ввід даних з клавіатури комп'ютера та друк результатів обчислень. У звіті до лабораторної роботи описати алгоритм, за яким побудована програма. При виводі даних обов'язково використати форматування.

16	Випадковим чином створити цілочисельний список. Визначити суму і кількість елементів, величина яких є меншою за середнє арифметичне елементів списку. Дописати значення суми і кількість елементів як елементи в кінець початкового списку та сформувати третій список з тих елементів, які дорівнюють середньому арифметичному елементів одержаного списку та початкового списку.

Роздруківка коду:

```
1 import random
2
3 try:
4     #Користувач вводить дані
5     minNum = input("Введіть мінімальне ціле значення числа у списку:
    ")
6     maxNum = input("І введіть максимальне ціле значення: ")
7     numLen = input("Введіть довжину списку (список повинен мати
    значення більше, ніж 1): ")
8
9     #Перевіряємо, чи користувач щось ввів
10    if minNum == "" or maxNum == "" or numLen == "":
11        raise ValueError("Ви не ввели один, два або жодного
    значення")
12
13    #Перетворюємо рядки на цілі числа
14    minNum = int(minNum)
```

```
15    maxNum = int(maxNum)
16    numLen = int(numLen)
17
18    #Перевіряємо, чи maxNum більший за minNum
19    if maxNum < minNum:
20        raise ValueError("Максимальне значення повинно бути більшим
    за мінімальне")
21
22    #Перевіряємо, чи довжина списку більша за 1
23    if numLen <= 1:
          raise ValueError("Довжина списку повинна бути більшою за 1")
24
25    #Створюється список, причому у випадковому порядку
26    lst = []
27    for _ in range(numLen):
28        lst_random = random.randint(minNum, maxNum)
29        lst.append(lst_random)
30
31    #Обчислюється середнє арифмет.знач. у списку
32    avg = sum(lst) / len(lst)
33
34    #Ці дані позначатимуть елементи, які менше за середнє
    арифмет.знач.
35    sum_below_avg = 0
36    count_below_avg = 0

37    #Копіюємо список, аби додавати елементи суми та лічильника в
    окремий список
38    lst2 = lst[:]
39
40    #Робота з елементами списку
41    for number in lst2:
42    #Якщо елемент менший за середнє, додаємо його до суми і
    збільшуємо лічильник
43        if number < avg:
44            sum_below_avg += number
45            count_below_avg += 1
46
47    #Додаємо значення суми і кількості до кінця початкового списку
48    lst2.append(sum_below_avg)
49    lst2.append(count_below_avg)
50
51    #Обчислюється середнє арифмет.знач. у списку з новими значеннями
52    avg1 = sum(lst2) / len(lst2)
53
54    #Створюємо третій список з тих елементів, які дорівнюють
    серед.арифмет.
55    lst3 = []
56    for x in lst2:
57        if x == avg:
58            lst3.append(x)
```

```
59
60      #Якщо для третього списку не знайшлося значень, то виведимо
    гарне повідомлення про відсутність значень
61      if lst3 == []:
62          lst3 = ("Немає значень")
63
64      #Виводимо результати на екран. Використовуються різні методи
    форматування залежно від типу виводу
65      print("Початковий список: {}".format(lst))
66      print(f"Середнє арифметичне: {avg:.2f}")
67      print("Сума елементів, які менші за середнє:
    {}".format(sum_below_avg))
68      print("Кількість елементів, які менші за середнє:
    {}".format(count_below_avg))
69      print("Початковий список із новими елементами: {}".format(lst2))
70      print(f"Середнє арифметичне списку з новими алементами:
    {avg:.2f}")
71      print("Третій список: {}".format(lst3))
72
73  except ValueError as text_of_error:
74      #Виводимо повідомлення про помилку
75      print("\nПомилка: {}".format(text_of_error))
```

Код має відмінний від типового для цього документа шрифт для зручного читання.

Знімок екрана тексту програми:

```
Введіть мінімальне ціле значення числа у списку: 1
І введіть максимальне ціле значення: 20
Введіть довжину списку (список повинен мати значення більше, ніж 1): 15
Початковий список: [8, 11, 11, 11, 1, 9, 7, 3, 20, 20, 15, 7, 14, 8, 20]
Середнє арифметичне: 11.00
Сума елементів, які менші за середнє: 43
Кількість елементів, які менші за середнє: 7
Початковий список із новими елементами: [8, 11, 11, 11, 1, 9, 7, 3, 20, 20, 15, 7, 14, 8, 20, 43, 7]
Середнє арифметичне списку з новими алементами:11.00
Третій список: [11, 11, 11]
```

Спробуймо також пропустити одне значення при вводі:

```
Введіть мінімальне ціле значення числа у списку: 1
І введіть максимальне ціле значення: 20
Введіть довжину списку (список повинен мати значення більше, ніж 1):

Помилка: Ви не ввели один, два або жодного значення
```

Алгоритм: Алгоритм виконання програми такий: імпортується модуль random, який дозволяє генерувати випадкові числа; потім використовується конструкція try-except, яка дозволяє відображати помилки, які допустив користувач при вводі значень; користувач вводить мінімальне і максимальне ціле значення числа у списку, а також довжину списку, яка повинна бути більшою за 1. Перевіряється, чи користувач щось ввів, якщо так, то чи це цілі числа, і щоб мінімальне значення не було більше за максимальне; якщо хоч щось було порушено, то користувач отримає текст помилки. Потім створюється порожній список і заповнюється випадковими цілими числами у заданому діапазоні, а на основі цього обчислюється середнє арифметичне значення елементів списку. Створюються дві змінні, яким присвоюються початкове значення 0: це будуть сума і кількість елементів початкового списку, які менші за середнє арифметичне; у новому списку програма проходиться по всіх елементах списку за допомогою циклу, і якщо елемент менший за середнє арифметичне, то додається до суми і збільшується лічильник на 1. Обчислюється середнє арифметичне значення елементів нового списку і зберігається у змінній, а потім створюється порожній список 3 і заповнюється елементами попереднього списку, які дорівнюють середньому арифметичному за допомогою циклу; якщо список 3 залишився порожнім, то присвоюється йому рядок “Немає значень”. Виводяться результати на екран за допомогою функції print(). Використовуються різні методи форматування залежно від типу виводу: метод format() для рядків і списків, і f-рядки для чисел з двома знаками після коми.

Завдання 2:

Відповідно до номера в списку вибрати індивідуальне завдання. Написати програму мовою Python. Забезпечити ввід даних з клавіатури комп’ютера та друк результатів обчислень. У звіті до лабораторної роботи описати алгоритм, за яким побудована програма. При виводі даних обов’язково використати форматування.

16	Згенерувати квадратну матрицю $A(m,m)$ розмірністю $m \times m$ з випадкових елементів, що є цілими числами. Сформувати нову матрицю, стовпцями якої будуть упорядковані за спаданням рядки початкової матриці.

Роздруківка коду:

```
1 import numpy as np
2
3 try:
4     #Користувач вводить дані
5     m = input("Дано: Матриця MxM, де m > 1:\nm = ")
6     minNum = input("Введіть мінімальне ціле значення числа у
    матриці: ")
7     maxNum = input("І введіть максимальне ціле значення: ")
```

```
8
9     # Перевіряємо, чи користувач щось ввів
10    if minNum == "" or maxNum == "" or m == "":
11        raise ValueError("Ви не ввели один, два або жодного
    значення")
12
13     #Перетворюємо рядки на цілі числа
14    minNum = int(minNum)
15    maxNum = int(maxNum)
16    m = int(m)
17
18    #Перевіряємо, чи maxNum більший за minNum
19    if maxNum < minNum:
20       raise ValueError("Максимальне значення повинно бути більшим
    за мінімальне")
21
22    #Перевіряємо, чи матриця більша за 1
23    if m <= 1:
24        raise ValueError("Розмір матриці повинний бути більшим за
    1")
25
26    #Генеруємо квадратну матрицю A(m,m)
27    A = np.random.randint(minNum, maxNum, (m, m))
28
29    #Виводимо початкову матрицю A
30   print("Початкова матриця A: \n{}".format(A))
31
32    #Сортуємо рядки матриці A за спаданням елементів
33    A_sorted = np.sort(A, axis=1)[:, ::-1]
34
35    #Транспонуємо матрицю A_sorted, щоб отримати нову матрицю B,
    стовпцями якої будуть упорядковані рядки A
36    B = A_sorted.T
37
38    # виводимо нову матрицю B
39    print("Нова матриця B: \n{}".format(B))
40
41 except ValueError as text_of_error:
42    #Виводимо повідомлення про помилку
43    print("\nПомилка: {}".format(text_of_error))
```

Знімок екрана тексту програми:

```
Дано: Матриця MxM, де m > 1:
m = 5
Введіть мінімальне ціле значення числа у матриці: -5
І введіть максимальне ціле значення: 20
Початкова матриця A:
[[-2 12  0 -4 11]
 [19 18 -2 12  2]
 [-1 -2 11 16  5]
 [ 6  2  8 17 -4]
 [ 2 15 -2 10  2]]
Нова матриця B:
[[12 19 16 17 15]
 [11 18 11  8 10]
 [ 0 12  5  6  2]
 [-2  2 -1  2  2]
 [-4 -2 -2 -4 -2]]
```

Алгоритм побудови програми: Алгоритм виконання програми такий: імпортується модуль numpy, який дозволяє працювати з багатовимірними масивами даних; потім використовується конструкція try-except, яка дозволяє відображати помилки, які допустив користувач при вводі значень; користувач вводить розмір квадратної матриці m, а також мінімальне і максимальне ціле значення елементів матриці. Перевіряється, чи користувач щось ввів, якщо так, то чи це цілі числа, і щоб мінімальне значення не було більше за максимальне; якщо хоч щось було порушено, то користувач отримає текст помилки. Потім генерується квадратна матриця A(m,m), яка повертає випадкові цілі числа у заданому діапазоні; виводиться початкова матриця A, і потім сортується рядки матриці A за спаданням елементів, який приймає параметр axis=1 для сортування по рядках, і зрізу [::-1] для інвертування порядку елементів. Транспонується матриця A, щоб отримати нову матрицю B, стовпцями якої будуть упорядковані рядки A. Виводиться нова матриця B .

Висновок: Виконавши цю лабораторну роботу, я зміг здобути відповідні навички в роботі з списками, кортежами, множинами і діапазонами в Python. Під час виконання лабораторної роботи проблем не виникало, а складність була в структуруванні коду та приведенні його до більш гарного вигляду.