Лабораторна робота №6

Кортежі та списки

Мета роботи

- Ознайомлення з кортежами та списками у мові Python.
- Алгоритм бінарного пошуку.

Короткі теоретичні відомості

1. Кортежі та списки

Кортежі та списки у мові Python є одними з основних типів даних. Їх використовують для групування разом декількох значень і позначають, як перелік елементів, розділених комами, укладений у квадратні – для списків і круглих – для кортежів дужки.

Кортеж та список – це впорядковані множини значень, що ідентифікуються індексом. Елементи списку та кортежу можуть бути любого типу. Впорядковані множини називають послідовностями.

1.1. Кортежі

Кортежі (англ. tuple) використовується для представлення незмінної послідовності різнорідних об'єктів. Вони зазвичай записуються в круглих дужках, але якщо неоднозначності не виникає, то дужки можна опустити.

```
>>> t = (2, 2.05, "Hello")
>>> t
(2, 2.0499999999999998, 'Hello')
>>> (a, b, c) = t
>>> print a, b, c
2 2.05 Hello
>>> z, y, x = t
>>> print z, y, x
2 2.05 Hello
>>> a=1
>>> b=2
>>> a,b=b,a
```

```
>>> print a,b
2 1
>>> x = 12,
>>> x
(12,)
```

Як видно з прикладу, кортеж може бути використаний і в лівій частині оператора присвоєння. Значення з кортежу в лівій частині оператора присвоєння зв'язуються з аналогічними елементами у правій частині.

1.2. Списки

У Python відсутні масиви в традиційному розумінні цього терміну. Замість них для зберігання однорідних (і не тільки) об'єктів використовуються списки.

На відміну від кортежів список не є незмінним (тобто ми можемо змінювати елементи, які містяться у списку). Кортеж - це незмінний (заморожений) список.

Існує декілька способів створення списків. Найпростіший з них: перерахувати елементи списку через кому в квадратних дужках:

```
>>> [10, 20, 30, 40]
[10, 20, 30, 40]
>>> ["one", "two", "three"]
['one', 'two', 'three']
```

Перший приклад – список чотирьох цілих чисел, а другий – список трьох стрічок. Елементи списків зовсім не обов'язково повинні бути одного типу. Наступний список містить стрічку, ціле і дробове числа і інший список:

```
>>> ["hello", 5, 2.0 [10, 20]]
['hello', 5, 2.0 [10, 20]]
```

Список, що є елементом іншого списку, називають вкладеним.

Список, який не містить жодного елементу, називають порожнім. Він позначається порожніми квадратними дужками ([]).

Ну, і нарешті, як і будь-які інші значення, списки можуть зберігатися в змінних:

```
numbers = [17, 123, 537]
empty = []
print numbers, empty
[17, 123, 537] []
```

1.3. Списки і індекси

Синтаксис звернення до елементів списку такий самий, як і при зверненні до елементів кортежу – використовуємо оператор індексування ([]).

```
>>> numbers[0]
17
>>> numbers[-1]
537
```

Індексом може бути будь-який вираз, що повертає ціле число, в тому числі від'ємне. Якщо індекс менше нуля, то відлік індексу буде розпочато з кінця списку: `numbers[-n] == numbers[len(li) - n`

Оператор побудови зрізу списку працює подібно, як і з кортежем:

```
>>> list = ['a', 'b', 'c', 'd', 'e', 'f']
>>> list[1:3]
['b', 'c']
>>> list[:4]
['a', 'b', 'c', 'd']
>>> list[3:]
['d', 'e', 'f']
>>> list[:]
['a', 'b', 'c', 'd', 'e', 'f']
```

Єдина відмінність полягає в тому, що елементи списків (на відміну від елементів котрежу) можна змінювати:

```
>>> numbers[1]= 5
>>> numbers
[17, 5, 537]
>>> numbers[1:]= [6, 7]
>>> numbers
[17, 6, 7]
>>> numbers[0], numbers[2]= 1, 2
>>> numbers
[1, 6, 2]
```

1.4. Довжина списку

Неважко здогадатися, що для обчислення довжини списку можна використати функцію `len()`. Зверніть увагу, що якщо список містить як елемент інший список, то цей вкладений список вважатиметься як один елемент. Це видно з наступного прикладу:

```
>>> mylist = [[1, 'one'], [2, 'two'], [3, 'three'], 'four', 5]
>>> len(mylist)
5
```

Якщо ви хочете у операторі for задати діапазон через кількість елементів в списку, то слід скористатися функцією `range` у поєднанні з функцією `len`:

```
>>> a = [`Linux', 'is', 'the', 'best', 'system']
>>> for i in range(len(a)):
... print i, a[i]     #Звертання до елементу списку по його індексу
...
0 Linux
1 is
2 the
3 best
4 system
```

1.5. Приклад пошуку максимального значення у списку

У якості прикладу використання списків та кортежів наведемо скрипт для пошуку максимального значення елементу у списку та його індексу. Ці значення збережемо у кортеж *max*

```
ls = [12, -4, 3, 15, 12]
max = (ls[0], 0)
for i in range(len(ls)):
     if ls[i] > max[0]:
          max = (ls[i], i)
print max
```

1.6. Вкладені (багатовимірні) списки

Список допускає, що його елементом може бути інший (вкладений) список:

```
>>> m = [1, 2, ['a', 'b', 'c'], 3, 8, 14]
>>> m[0]
1
>>> m[1]
2
>>> m[2]
['a', 'b', 'c']
>>> m[3]
```

```
3
```

Зверніть увагу, що m[2] - це список ['a', 'b', 'c'] цілком.

Як звернеться до одного з елементів цього вкладеного списку? Потрібно вказати ще один, додатковий індекс:

```
>>> m
[1, 2, ['a', 'b', 'c'], 3, 8, 14]
>>> m[2]
['a', 'b', 'c']
>>> m[2][0]
'a'
>>> m[2][1]
'b'
>>> m[2][:2]
['a', 'b']
```

1.7. Дії зі списками

Списки допускають два типи дій - конкатенацію і розмноження (мультиплікацію). Конкатенация позначає склейку списків і здійснюється наступним чином:

```
>>> s1 = [1, 2, 3, 4]
>>> s2 = [10, 20, 30, 40]
>>> s3 = ['a', 'b', 'c']
>>> sk = s1 + s2 + s3 + [2, 2]
>>> sk
[1, 2, 3, 4, 10, 20, 30, 40, 'a', 'b', 'c', 2, 2]
```

Розмноження списків здійснюється за допомогою знака множення:

```
>>> s1
[1, 2, 3, 4]
>>> sm = s1 * 4
>>> sm
[1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
```

Для перетворення кортежу в список є функція list, для зворотної операції - tuple.

```
>>> ls = [1, 2, 3]
>>> tup = tuple(ls)
>>> tup
(1, 2, 3)
```

```
>>> ls2 = list(tup)
>>> ls2
[1, 2, 3]
```

Завдання та порядок виконання роботи

1. Ознайомитися з наведеними вище теоретичними відомостями.

2. Виконати приклади, які приводяться в теоретичних відомостях.

3. Виконати наступні завдання:

   3.1. Написати скрипт, який:

   - на початку, у вигляді коментаря, буде містити назву курсу та номер лабораторної роботи, а також ваше ім'я та прізвище, та номер Вашої заліковки
   - у першому рядку буде виводити назву курсу та номер лабораторної роботи
   - у другому – буде виводити ваше ім'я та прізвище, а також номер Вашої заліковки.
   - написати скрипт за варіантом (не використовувати спеціальні вбудовані методи по роботі зі списками):

   1) Знайти мінімальне та максимальне значення у списку довільної довжини. Створити кортеж, який буде містити ці значення та матиме наступний вигляд ((min_value, min_index), (max_value, max_index))

   2) Знайти середнє арифметичне додатних та від'ємних значення у списку довільної довжини. Створити кортеж, який буде містити ці значення та матиме наступний вигляд (sum_positive, sum_negative)

   3) Дани два вектори a = (ax, ay, az) та b = (bx, by, bz). Знайти їх суму, скалярний та векторний добуток, а також норму кожного з них і кут між ними.

   4) Знайти мінімальне та максимальне значення у списку довільної довжини та обміняти їх місцями. У результаті має бути кортеж, який містить список з обміняними місцями максимальним та мінімальним елементами

   5) Дано список елементи якого відсортовані за зростанням. Використовуючи алгоритм бінарного пошуку дізнатись, чи є у списку заданий елемент. У результаті створити кортеж який містить шуканий елемент та його індекс (value, index). Якщо такого елементу нема, то у значення індексу має дорівнювати -1.

   6) Дано список елементи якого відсортовані за спаданням. Використовуючи алгоритм бінарного пошуку дізнатись, чи є у списку заданий елемент. У результаті створити кортеж який містить шуканий елемент та його індекс (value, index). Якщо такого елементу нема, то у значення індексу має дорівнювати -1.

7) Прибрати зі списку усі дублі елементів. У результаті має бути отриманий кортеж без дублів. Наприклад [1, -2, 3, 1, 4, -2] -> (1, -2, 3, 4)

8) Дано два списки. Найти їх перетин – ті елементи які є в обох списках, і зберегти їх до кортежу. Наприклад X = [1, 4, -3, 5] і Y = [-2, 1, 3]. Тоді X∩Y=(1, 3)

9) Дано два списки. Найти їх симетричну різницю – ті елементи з першого списку яких нема у другому та ті елементи з другого списку яких нема у першому, і зберегти їх до кортежу. Наприклад X = [1, 4, -3, 5] і Y = [-2, 1, 3]. Тоді XΔY=(4, -2, 3, -3, 5)

10) Дано два списки. Найти їх різницю – ті елементи з першого списку яких нема у другому, і зберегти їх до кортежу. Наприклад X = [1, 4, -3, 5] і Y = [-2, 1, 3]. Тоді X\Y=(4, -3, 5)

11) Дано два кортежі A і B. Перевірити чи один з них входить в інший. Якщо так, то створити кортеж, якій містить індекс, починаючи з якого один кортеж входить в інший та кортеж, що входить до іншого кортежу. Наприклад, якщо A=(2, 3, -4, 5, 8, 1) і B=(-4, 5, 8) то у результаті має бути res=(2, (-4, 5, 8))

12) Використовуючи метод бінарного пошуку знайти розв'язок рівняння $f(x)=x^3-x-2$ з точністю eps=0.0001 на проміжку [1, 2]. У результаті отримати кортеж, у якому буде міститись розв'язок та кількість виконаних ітерацій.

13) Дано список який містить набір результати деяких вимірювань одного значення (наприклад прискорення вільного падіння). Знайти математичне очкування $M[X] = \frac{1}{n}\sum_{i=1}^{n} x_i$ та дисперсію $D = \frac{\sum_{i=1}^{n} X_i^2 - \frac{\left(\sum_{i=1}^{n} X_i\right)^2}{n}}{n-1}$. У результаті отримати кортеж який містить значення (M, D).

14) Дано список який містить набір результати деяких вимірювань температури (як додатні, так і від'ємні значення). Знайти значення температури та її індекс, яка була ближча за все до 0. У результаті вивести кортеж, який містить значення температури найближчої до 0 та її індекс у списку.

15) Використовуючи метод бінарного пошуку знайти розв'язок рівняння $f(x)= 5^x - 6x -3$ з точністю eps=0.0001 на проміжку [1, 2]. У результаті отримати кортеж, у якому буде міститись розв'язок та кількість виконаних ітерацій.

16) Використовуючи метод бінарного пошуку знайти розв'язок рівняння $f(x)= e^x – (x + 2)$ з точністю eps=0.0001 на проміжку [0, 2]. У результаті отримати кортеж, у якому буде міститись розв'язок та кількість виконаних ітерацій.

4. Зберегти скрипт у файл з наступною назвою <Surname>_Task6.py (наприклад Rodionov_Task6.py)

5. Запустити даний скрипт за допомогою інтерпретатора та переконатись у його працездатності та правильності результатів.

6. Спробувати здати та захистити лабораторну роботу у викладача практичних занять

Література

1. A Byte of Python (Russian) (http://wombat.org.ua/AByteOfPython/#id14)
2. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с..
3. Computational Physics with Python by M. Newman (http://www-personal.umich.edu/~mejn/computational-physics/)

Інтернет посилання

- http://ru.wikipedia.org/wiki/Python
- MIT: Introduction to Computer Science and Programming
- http://python.org