Дисципліна «Основи програмування на Python»

Циклічні алгоритми. Конструкція for

Медведєв Д.Г.

Зараз ми розглядаємо одну з двох основних циклічних конструкції мови Python – операторів, які повторюють дію знову і знову. Перший з них, оператор `while`, пропонує спосіб написання універсальних циклів. Другий, оператор `for`, призначений для проходу по елементам у послідовності або в іншому ітеруючому об’єкті та виконання коду блоку для кожного елемента.

Цикл `for` є універсальним ітератором у Python: він може проходити по елементам у будь-якій упорядкованій послідовності або в іншому ітерованому об’єкті. Оператор працює з рядками, списками, кортежами і іншими вбудованих ітерованими об'єктами, а також на нових об'єктах, визначених користувачем.

Цикл `for` починається з рядку заголовка, де виявляється ціль (або цілі) присвоєння разом з об'єктом, за яким потрібно виконати прохід. Після заголовка знаходиться блок операторів (звичайно з відступами), який необхідно повторити:

```
for ціль in об'єкт:
    оператори
else:
    оператори
```

Коли Python запускає цикл `for`, він присвоює цілі елементи ітерованого об'єкта по черзі і виконується для кожного тіло циклу. Всередині тіла циклу ціль присвоєння зазвичай використовується для посилань на поточний елемент у послідовності.

Ім'я, яке застосовується як ціль присвоєння в рядку заголовка `for`, зазвичай є (можливо, новою) змінною всередині області видимості, де знаходиться оператор for. Ім'я не відрізняється якоюсь унікальністю; його навіть можна змінити всередині тіла циклу, але воно буде автоматично встановлено в наступний елемент послідовності, коли управління знову повернеться в початок циклу. Після циклу ця змінна, як правило, продовжує посилатись на останній відвідуваний елемент, який буде останнім елементом у послідовності, якщо тільки не вийшов вихід із циклу через оператора `break`.

Оператор `for` також підтримує необов’язковий блок `else`, який працює точно так, як у циклі `while` – він виконується, якщо вихід із циклу здійснюється без допомоги оператора `break` (тобто коли були пройдені всі елементи послідовності

Як приклад будемо надавати змінній x кожен з трьох елементів по черзі, зліва направо, і для кожного з них буде виконуватися оператор `print`. Всередині оператора `print` (тіло циклу) змінна x посилається на поточний елемент у списку:

```
for x in ['spam', 'second', 'third']:
    print(x, end=' ')
# spam second third
```

В циклі `for` працює ,будь-яка послідовність, він представляє собою універсальний інструмент. Скажімо, for працює на рядках і кортежах:

```
S = "lumberjack"
T = ("and", "I'm", "okay")
for x in S:
    print (x, end=' ')
# l u m b e r j a c k
for x in T:
    print(x, end=' ')
# and I'm okay
```

Кортежі в циклах також стають корисними при ітераціях одразу по ключах і значеннях у словниках із застосуванням методу `items` замість проходу в циклі по ключах та індексаціях для вилучення значень вручну:

```
D = {'a' : 1, 'b' : 2, 'c' : 3}
for (key, value) in D.items():
    print (key, '=>' , value)
# a => 1
# c => 3
# b => 2
```