Урок №9. Истинная мощь цикла for. Кортежи. Сортировка.

План урока

1. Цикл for ... in ... по элементам списка или строки.
2. Контейнеры.
3. Кортежи.
4. Множественное присваивание.
5. Сортировка пузырьком.

Аннотация

В уроке вводится важная конструкция "`for` ... `in` (список или строка)" для прохода по контейнерам. Вводится ещё один контейнер — кортеж (`tuple`), что позволяет, в свою очередь, ввести множественное присваивание и немедленно применить его в реализации классического алгоритма — сортировки пузырьком.

Повторение

Задача «Супы (https://contest.yandex.ru/contest/3002/problems/A/)».

Пока что мы использовали цикл `for` для того, чтобы перебирать числа из заданного диапазона. Однако у этой синтаксической конструкции есть и другие возможности. С помощью тех же ключевых слов `for` и `in` можно перебрать элементы списка — не индексы, а сами значения:

```
months = ['январь', 'февраль', 'март', 'апрель', 'май', 'июнь',
          'июль', 'август', 'сентябрь', 'октябрь', 'ноябрь', 'декабрь']
for month in months:
    print(month)
```

Обратите внимание, что здесь не используется `range`, а после слова `in` сразу пишется список, элементы которого мы перебираем в этом цикле. Переменная `month` также присутствует, но принимает не численные значения, а значения элементов списка `months` (в данном случае — строки).

Конечно, если нам требуется использовать **индексы** элементов, то придётся перебирать индексы с помощью обычного цикла `for i in range(len(months))`.

Сравните:

```
months = ['январь', 'февраль', 'март', 'апрель', 'май', 'июнь',
          'июль', 'август', 'сентябрь', 'октябрь', 'ноябрь', 'декабрь']
for i in range(len(months)):
    print('Месяц номер', i + 1, 'называется', months[i])
```

Увидели разницу?

Цикл `for` может перебрать все значения не только в любом списке, но и в других объектах, которые можно разделить на отдельные части (причём таких частей в разных объектах того же типа может быть сколько угодно). Такие объекты называются **контейнерами**. В частности, помимо списков, контейнерами являются строки: ведь строка состоит из отдельных символов, к которым можно обратиться по индексу. Стало быть, можно и перебрать символы строки в цикле `for`:

```
for letter in 'тридцать.символов.по.вертикали':
    print(letter)
```

Тогда что же такое `range(n)`?

Уже ясно, что это не неотъемлемая часть цикла `for`. Похоже на функцию, которая выдаёт определённый список. Это не совсем так, но `range(n)` ведёт себя очень похоже на список. Пока он используется прямо в цикле `for`, можно считать, что, например, `range(5)` — это список [0, 1, 2, 3, 4], и ничего больше.

```
for i in [0, 1, 2, 3, 4]:
    print(i)
for i in range(5):
    print(i)
# Разницы не видно.
```

Задачи:

- **Отбор (https://contest.yandex.ru/contest/3002/problems/B/)**,
- **Инвестиционный фонд (https://contest.yandex.ru/contest/3002/problems/C/)** ,

Необязательные задачи:

- **Ххооллоодд (https://contest.yandex.ru/contest/4820/problems/A/)**,
- **Считать и вывести таблицу — 3 (https://contest.yandex.ru/contest/4820/problems/B/)** +,
- **Окно (https://contest.yandex.ru/contest/4820/problems/C/)**.

Рассмотрим ещё один контейнер. Это - `tuple` (читается «тюпл» или «тьюпл», а переводится как «кортеж»).

Кортежи очень похожи на списки, только вместо квадратных в них используются круглые скобки (причём эти круглые скобки часто можно пропускать):

```
card = ('7', 'пик')                  # кортеж из двух элементов; тип элементов может быть любой
empty = ()                           # пустой кортеж (из 0 элементов)
t = (18,)                            # кортеж из 1 элемента - запятая, чтобы отличить от обычных скобок
print(len(card), card[0], card + t) # длина, значение отдельного элемента, сложение - как у списков
```

Сравнивать между собой кортежи тоже можно.

Но есть и отличия.

Важнейшее техническое отличие — **неизменяемость**. Как и к строке, к кортежу нельзя добавить элемент методом `append`, а существующий элемент нельзя изменить, обратившись к нему по индексу. Это выглядит недостатком, но в дальнейшем мы поймём, что у кортежей есть и преимущества.

Есть и семантическое (то есть смысловое) отличие. Если списки предназначены скорее для объединения неопределённого количества однородных сущностей, то кортеж — это быстрый способ объединить под одним именем несколько разнородных объектов, имеющих различный смысл.

Так, в примере выше кортеж `card` состоит из двух элементов, означающих достоинство карты и её масть.

Именно благодаря кортежам работает красивая особенность Питона — **множественное присваивание**.

Как известно, по левую сторону от знака присваивания `=` должно стоять имя переменной либо имя списка с индексом (или несколькими индексами). Они указывают, куда можно «положить» значение, записанное справа от знака присваивания. Однако слева от знака присваивания можно записать ещё и кортеж из таких обозначений (грубо говоря, имён переменных), а справа — кортеж из значений, которые следует в них поместить. Значения справа указываются в том же порядке, что и переменные слева (здесь скобки вокруг кортежа не обязательны):

```
n, s = 10, 'hello'
# то же самое, что
n = 10
s = 'hello'
```

В примере выше мы изготовили кортеж, стоящий справа от `=`, прямо на этой же строчке. Но можно заготовить его и заранее:

```
cards = [('7', 'пик'), ('Д', 'треф'), ('Т', 'пик')]
value, suit = cards[0]
print('Достоинство карты:', value)
print('Масть карты:', suit)
```

Самое приятное: сначала вычисляются все значения справа, и лишь затем они кладутся в левую часть оператора присваивания. Поэтому можно, например, поменять местами значения переменных a и b, написав: `a, b = b, a`.

```
a, b = 1, 2  # теперь a == 1 and b == 2
a, b = b, a  # теперь a == 2 and b == 1
# Без множественного присваивания потребуется дополнительная переменная:
t = a
a = b
b = t  # теперь a == 1 and b == 2
# А вот так менять их местами нельзя:
a = b
b = a
# теперь a == b == 2
```

Пример ниже выведет «1 2 3». Убедитесь, что вы понимаете, почему так.

```
# кручу-верчу
a, b, c = 3, 2, 1
b, a, c = c, a, b
print(b, c, a)
```

Заметьте, что, например, вычисление чисел Фибоначчи теперь приобретает волшебную краткость:

```
n = int(input())
f1, f2 = 0, 1
for i in range(n):
    print(f2)
    f1, f2 = f2, f1 + f2
```

Задача «1984 (https://contest.yandex.ru/contest/3002/problems/D/)».

Итак, у нас есть удобный способом поменять местами значения двух переменных. Теперь рассмотрим алгоритм, в котором эта операция играет важную роль.

Часто бывает нужно, чтобы данные не просто содержались в списке, а были отсортированы (например, по возрастанию), то есть чтобы каждый следующий элемент списка был не меньше предыдущего. В качестве данных могут выступать числа или строки. Скажем, отсортированный список `[4,1,9,3,1]` примет вид `[1,1,3,4,9]`. Конечно, для этого есть стандартные функции и методы — но как они работают?

Классический алгоритм сортировки — **сортировка пузырьком** (по науке - **сортировка обменом**). Она называется так потому, что элементы последовательно «всплывают» (отправляются в конец списка) — как пузырьки воздуха в воде. Сначала всплывает самый большой элемент, за ним — следующий по старшинству и т. д. Для этого мы сравниваем по очереди все соседние пары и, при необходимости, меняем элементы местами, ставя больший элемент на более старшее место. Идею объясняет венгерский народный танец (https://www.youtube.com/watch?v=lyZQPjUT5B4), а полный код программы, которая считывает, сортирует и выводит список, выглядит, например, так:

```
n = int(input())    # количество элементов
a = []
for i in range(n):  # считываем элементы списка
    a.append(int(input()))
# Сортировка пузырьком:
for i in range(n - 1):
    for j in range(n - 1 - i):
        if a[j] > a[j + 1]:
            a[j], a[j + 1] = a[j + 1], a[j]
print(a)
```

Задачи в классе:

- **Супы (https://contest.yandex.ru/contest/3002/problems/A/)**
- **Отбор (https://contest.yandex.ru/contest/3002/problems/B/)**
- **Инвестиционный фонд (https://contest.yandex.ru/contest/3002/problems/C/)**
- **1984 (https://contest.yandex.ru/contest/3002/problems/D/)**
- **Сортировка по алфавиту (https://contest.yandex.ru/contest/3002/problems/E/)**

Домашние задачи:

- **Числа Трибоначчи (https://contest.yandex.ru/contest/3006/problems/A/)**
- **bf-- (https://contest.yandex.ru/contest/3006/problems/B/)**

Дополнительные задачи:

- **Ххооллоодд (https://contest.yandex.ru/contest/4820/problems/A/)**
- **Считать и вывести таблицу — 3 (https://contest.yandex.ru/contest/4820/problems/B/) +**
- **Окно (https://contest.yandex.ru/contest/4820/problems/C/)**
- **Сортировка в обратном порядке (https://contest.yandex.ru/contest/4820/problems/D/)**
- **Сортировка по длине (https://contest.yandex.ru/contest/4820/problems/E/)**
- **Крупные буквы (https://contest.yandex.ru/contest/4820/problems/F/)**
- **Разные разности (https://contest.yandex.ru/contest/4820/problems/G/)**
- **Проверка блокчейна (https://contest.yandex.ru/contest/4820/problems/H/)**
- **bf (https://contest.yandex.ru/contest/4820/problems/I/)**