Урок 4  
Цикл for и кортежи

короткая линия

# План урока

1. Кортежи
2. Функция range
3. Цикл for
4. Вложенные циклы

# Кортежи

Кроме строк, которые по сути являются последовательностями отдельных символов, в языке Python существует такой тип данных, как кортеж. Кортежи применяются не так часто, как списки (читай, массивы), которые мы будем проходить позже, однако это достаточно важная для языка структура данных.

Кортежи могут состоять из произвольных элементов (т. е. кортеж может содержать как числа, так и строки, а также любые другие типы) и являются неизменяемыми. Это значит, что после присваивания вы не сможете поменять ни один элемент в кортеже. Кортеж объявляется так:

|  |
| --- |
| a = (10, 20, 13) |

Если кортеж состоит из одного элемента, его следует объявлять так (иначе интерпретатор воспримет элемент как одиночную строку):

|  |
| --- |
| a = (37,) |

Также вы можете трансформировать объекты (например, строки) в кортежи. Делается это с помощью функции tuple():

|  |
| --- |
| a = tuple(**"Hello world!"**) |

Кортежи можно складывать и умножать на число. Посмотрите, что произойдет, если выполнить подобные команды. У каких объектов вы видели похожее поведение?

Чтобы обратиться к определенному элементу кортежа, используются квадратные скобки с указанием индекса в них. Например:

|  |
| --- |
| a = (10, 20, 13) b = a[0] *# в переменной b хранится значение 10* |

**Важно:** в Python индексация всегда начинается с нуля.

Так как кортежи принимают в себя любые типы данных, элементом кортежа может также быть и другой кортеж.

|  |
| --- |
| a = (20, 10, 13, (1, 2, 3)) b = a[3][1] |

Обратите внимание на усложненную индексацию: в этом примере мы обращаемся к четвертому элементу кортежа a и ко второму элементу полученного кортежа.

Существует операция, которая называется распаковка кортежа. Допустим, у нас есть кортеж, содержащий информацию об ученике:

|  |
| --- |
| student = (**"Шейх"**, **"Мусса"**, **"Python"**, 40) name, surname, language, age = student |

Мы перенесли (“распаковали”) значения кортежа, поместив каждое из них в соответствующие переменные. Посмотрите, что произойдет, если количество переменных будет отличаться от длины кортежа.

**Интересно:**

1. Кортежи позволяют быстро менять значение двух и более переменных:

a, b = b, a

2) Вы можете инициализировать переменные на одной строке:

n1, n2, n3 = 10, 20, int(input())

# Функция range

Функция range() используется для генерации ряда целых чисел в заданном диапазоне. В общем виде функция выглядит так: range(from, to, step), где from – это первое значение диапазона (целое число), to – граница диапазона (это число не включается!), а step – интервал между числами диапазона. В такой конфигурации получаем, что: from < to, step > 0.

Возможна и установка убывающего ряда. Тогда получаем: from > to, step < 0.

print(tuple(range(1, 20)))

print(tuple(range(-100, -120, -1)))

Несколько деталей:

1. Функция range возвращает объект range, поэтому, чтобы вывести данные в том виде, в каком мы желаем, следует преобразовать этот объект в кортеж чисел.
2. Если значение from равно нулю, его можно не указывать и просто писать, например, range(10).

Пользуясь базовыми примерами, решите задачи ниже:

**Задача “Вывести диапазон”**

**Задача “Вывести четные”**

# Цикл for

Циклы – наиболее часто употребляемые конструкции в любом языке. Они позволяют анализировать информацию, перебирать значения, сортировать и т. п. На сегодняшнем занятии мы поговорим о цикле for. Этот цикл удобен, когда мы точно знаем, сколько итераций (повторений определенного блока команд) следует сделать. Рассмотрим, как он используется на примере:

|  |
| --- |
| **for** i **in** range(10):  print(i) |

Данный фрагмент просто выводит все числа от нуля до девяти. Можно заметить несколько особенностей:

1. Присутствует двоеточие, и блок выполняемого кода помещается в отступы, аналогично с условными конструкциями.
2. Вводится переменная i. На каждой итерации мы перебираем одно из значений диапазона range(10), и это значение заносится в переменную. Таким образом, мы можем обращаться к i внутри блока, относящегося к циклу: например, в функции print().

**Задача “Убывающий ряд”**

**Задача “Цикл с условием”**

**Задача “Сумма ряда”**

**Задача “18! и не только”**

Однако перебирать можно не только значения из диапазона, но и проходиться по элементам кортежа или по символам строки:

|  |
| --- |
| winter\_months = (**"декабрь"**, **"январь"**, **"февраль"**) **for** i **in** winter\_months:  print(i, **"- это зимний месяц"**) |

Данный код позволяет нам пройтись по каждому элементу кортежа и вывести его.

**Задача “Цвета радуги”**

**Задача “Ограниченный бюджет”**

Как и с условными операторами, вы можете создавать циклы произвольной вложенности:

|  |
| --- |
| colors = (**"red"**, **"green"**, **"yellow"**) fruits = (**"apple"**, **"pear"**) **for** i **in** colors:  **for** j **in** fruits:  print(i, j) |

**Домашнее задание:**

**Задача “Сумма квадратов”**

**Задача “12 месяцев”**

**Задача “Коровы”**

**Задача “Числа Фибоначчи”**

**Задача “Проще некуда”**

**Задача “Таблица Пифагора”**