**Лекция 4.**

**Списки**

Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов.

Чтобы использовать списки, их нужно создать.

**>>> s = [] # Пустой список**

**>>> l = ['s', 'p', ['isok'], 2]**

**>>> s**

**[]**

**>>> l**

**['s', 'p', ['isok'], 2]**

Как видно из примера, список может содержать любое количество любых объектов (в том числе и вложенные списки), или не содержать ничего.

Для списков доступны основные встроенные функции, а также методы списков.

**Таблица "методы списков"**

| **Метод** | **Что делает** |
| --- | --- |
| **list.append**(x) | Добавляет элемент в конец списка |
| **list.extend**(L) | Расширяет список list, добавляя в конец все элементы списка L |
| **list.insert**(i, x) | Вставляет на i-ый элемент значение x |
| **list.remove**(x) | Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует |
| **list.pop**([i]) | Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент |
| **list.index**(x, [start [, end]]) | Возвращает положение первого элемента со значением x (при этом поиск ведется от start до end) |
| **list.count**(x) | Возвращает количество элементов со значением x |
| **list.sort**([key=функция]) | Сортирует список на основе функции |
| **list.reverse**() | Разворачивает список |
| **list.copy**() | Поверхностная копия списка |
| **list.clear**() | Очищает список |

Методы списков изменяют сам список, а потому результат выполнения не нужно записывать в эту переменную.

**>>> l = [1, 2, 3, 5, 7]**

**>>> l.sort()**

**>>> l**

**[1, 2, 3, 5, 7]**

**Словари**

**Словари в Python** - неупорядоченные коллекции произвольных объектов с доступом по ключу. Их иногда ещё называют ассоциативными массивами или хеш-таблицами.

Чтобы работать со словарём, его нужно создать. Создать его можно несколькими способами:

**>>> d = {}**

**>>> d {}**

**>>> d = {'dict': 1, 'dictionary': 2}**

**>>> d**

**{'dict': 1, 'dictionary': 2}**

Также можно использовать функцию dict

**>>> d = dict(short='dict', long='dictionary')**

**>>> d**

**{'short': 'dict', 'long': 'dictionary'}**

**Таблица "методы словарей"**

| **Метод** | **Что делает** |
| --- | --- |
| **dict.clear**() | очищает словарь |
| **dict.copy**() | возвращает копию словаря |
| **dict.fromkeys**(seq[, value]) | создает словарь с ключами из seq и значением value (по умолчанию None) |
| **dict.get**(key[, default]) | возвращает значение ключа, но если его нет, не бросает исключение, а возвращает default (по умолчанию None) |
| **dict.items**() | возвращает пары (ключ, значение) |
| **dict.keys**() | возвращает ключи в словаре |
| **dict.pop**(key[, default]) | удаляет ключ и возвращает значение. Если ключа нет, возвращает default (по умолчанию бросает исключение) |
| **dict.popitem**() | удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение KeyError. Помните, что словари неупорядочены |
| **dict.setdefault**(key[, default]) | возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ с значением default (по умолчанию None) |
| **dict.update**([other]) | обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает None |
| **dict.values**() | возвращает значения в словар |

**Кортежи**

Кортеж, по сути - неизменяемый список.

Основное назначение кортежей – экономия памяти и защита от дурака.

Создание кортежей:

**>>> a = tuple() # С помощью встроенной функции tuple()**

**>>> a**

**()**

**>>> a = () # С помощью литерала кортежа**

**>>> a**

**()**

Работа с кортежами:

Все операции над списками, не изменяющие список (сложение, умножение на число, методы index() и count() и некоторые другие операции). Можно также по-разному менять элементы местами и так далее.

Например, гордость программистов на python - поменять местами значения двух переменных:

**a, b = b, a**