**Основные типы данных в Python**

Каждое значение в Python имеет тип. Поскольку всё в Python — объекты, **типы являются классами, а значения — экземплярами (объектами) этих классов**.

**Числа**

Целые числа, числа с плавающей запятой и комплексные числа относятся к группе чисел. В Python они представлены **классами int, float и complex**.

Мы можем использовать функцию **type(), чтобы узнать класс переменной** или значения, и функцию **isinstance() для проверки принадлежности объекта определённому классу:**

>>> a = 5

>>>print(a, "is of type", type(a))

5 is of type <class 'int'>

>>> a = 2.0

>>>print(a, "is of type", type(a))

2.0 is of type <class 'float'>

>>> a = 1+2j

>>>print(a, "is complex number?", isinstance(1+2j, complex))

(1+2j) iscomplexnumber? True

**Целые числа могут быть любой длины**, они ограничиваются лишь доступной памятью.

**Числа с плавающей запятой имеют** [**ограниченную точность**](https://goo.gl/cBH7Jw). Визуально разницу между целым числом и числом с плавающей запятой можно заметить в консоли по наличию точки: 1 — целое число, 1.0 — с плавающей запятой.

Комплексные числа записываются в форме x+yj, где x — действительная часть числа, а y — мнимая. Вот несколько примеров:

>>> a = 1234567890123456789

>>> a

1234567890123456789

>>> b = 0.1234567890123456789

>>> b

0.12345678901234568

>>> c = 1+2j

>>> c

(1+2j)

Обратите внимание, что значение переменной b было усечено.

**Списки**

Список представляет собой упорядоченную последовательность элементов. Он очень гибкий и является одним из самых используемых типов в Python. Элементы списка не обязательно должны быть одного типа.

Объявить список довольно просто. Внутрь квадратных скобок помещаются элементы списка, разделённые запятой:

>>> a = [1, 2.2, 'python']

Мы можем использовать оператор [] для извлечения элемента (такая операция называется «доступ по индексу») или диапазона элементов (такая операция назвается «извлечение среза») из списка. В Python индексация начинается с нуля:

>>> a = [5,10,15,20,25,30,35,40]

>>>print("a[2] =", a[2])

a[2] = 15

>>>print("a[0:3] =", a[0:3])

a[0:3] = [5, 10, 15]

>>>print("a[5:] =", a[5:])

a[5:] = [30, 35, 40]

Списки являются изменяемым типом, т.е. значения его элементов можно изменить:

>>> a = [1,2,3]

>>> a[2] = 4

>>> a

[1, 2, 4]

**Кортежи**

Так же как и список, кортеж (tuple) является упорядоченной последовательностью элементов. Вся разница заключается в том, что кортежи неизменяемы.

Кортежи используются для защиты данных от перезаписи и обычно работают быстрее, чем списки, т.к. их нельзя изменять.

Для создания кортежа нужно поместить внутрь круглых скобок элементы, разделённые запятой:

>>> t = (5,'program', 1+3j)

Мы можем использовать оператор извлечения среза [] для извлечения элементов, но мы не можем менять их значения:

>>> t = (5,'program', 1+3j)

>>>print("t[1] =", t[1])

t[1] = program

>>>print("t[0:3] =", t[0:3])

t[0:3] = (5, 'program', (1+3j))

# Приводит к ошибке, т.к.

# кортежи неизменяемы

>>> t[0] = 10

**Строки**

Строка представляет собой последовательность символов. Мы можем использовать одинарные или двойные кавычки для создания строки.  Многострочные строки можно обозначить тройными кавычками, ''' или """:

>>> s = "Простая строка"

>>> s = '''многострочная

строка'''

Как и в случае со списками и кортежами, мы можем использовать оператор [] и со строками. Стоит отметить, что строки в Python относятся к категории неизменяемых последовательностей, то есть все функции и методы могут лишь создавать новую строку.

**Множества**

Множество является неупорядоченной уникализированной последовательностью. Объявляется множество с помощью элементов, разделённых запятой, внутри фигурных скобок:

>>> a = {5,2,3,1,4}

# вывод переменной множества

>>>print("a =", a)

a = {1, 2, 3, 4, 5}

# тип данных переменной а

>>>print(type(a))

<class 'set'>

Над множествами можно выполнять такие операции, как объединение и пересечение. Т.к. элементы в множестве должны быть уникальны, они автоматически удаляют дубликаты:

>>> a = {1,2,2,3,3,3}

>>> a

{1, 2, 3}

Поскольку множество является неупорядоченной последовательностью, оператор извлечения среза здесь не работает:

>>> a = {1,2,3}

>>>a[1]

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'set' object does not support indexing

**Словари**

Словари — неупорядоченные наборы пар ключ-значение.

Они используются, когда нужно сопоставить каждому из ключей значение и иметь возможность быстро получать доступ к значению, зная ключ. В других языках словари обычно называются map, hash или object. Словари оптимизированы для извлечения данных. Чтобы извлечь значение, нужно знать ключ.

Словарь объявляется парами элементов в форме ключ:значение, заключенными в фигурные скобки:

>>> d = {1:'value', 'key':2}

>>>type(d)

<class 'dict'>

Значение может быть любого типа, а вот ключ — только неизменяемого.

Мы используем ключ, чтобы получить соответствующее ему значение. Но не наоборот:

>>> d = {1:'value', 'key':2}

>>>print("d[1] =", d[1]);

d[1] = value

>>>print("d['key'] =", d['key']);

d['key'] = 2

# Приводит к ошибке

>>>print("d[2] =", d[2]);

**Преобразование типов данных**

Мы можем преобразовывать значения из одного типа в другой с помощью таких функций, как int(), float(), str() и т.д.

>>>float(5)

5.0

При преобразовании числа с плавающей запятой в целое будет утеряна часть после запятой:

>>>int(10.6)

10

>>>int(-10.6)

-10

Для преобразования из/в строку должны использоваться совместимые значения:

>>>float('2.5')

2.5

>>>str(25)

'25'

>>>int('1p')

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ValueError: invalid literal for int() with base 10: '1p'

Можно даже преобразовывать одну последовательность в другую:

>>>set([1,2,3])

{1, 2, 3}

>>>tuple({5,6,7})

(5, 6, 7)

>>>list('hello')

['h', 'e', 'l', 'l', 'o']

Прим. перев. Для преобразования списка из символов обратно в строку нельзя вызвать str(список), так как в результате мы получим строковое представление списка (наподобие того, что мы видим, когда выводим список на экран). Вместо этого нужно сделать следующее:

''.join(['h', 'e', 'l', 'l', 'o'])

Для преобразования в словарь каждый элемент последовательности должен быть парой:

>>>dict([[1,2],[3,4]])

{1: 2, 3: 4}

>>>dict([(3,26),(4,44)])

{3: 26, 4: 44}