# Составные типы данных

В Python типы данных можно разделить на встроенные в интерпретатор *(built-in*) и не встроенные, которые можно использовать при импортировании соответствующих модулей.

**К основным встроенным типам относятся**:

1. *None* (неопределенное значение переменной)
2. Логические переменные (*Boolean Type*)
3. Числа (*Numeric Type*)
	1. ***int* – целое число**
	2. ***float* – число с плавающей точкой**
	3. *complex* – комплексное число
4. Списки (*Sequence Type*)
	1. ***list* – список**
	2. ***tuple* – кортеж**
	3. ***range* – диапазон**
5. Строки (*Text Sequence Type* )
	1. ***str***
6. Бинарные списки (*Binary Sequence Types*)
	1. ***bytes* – байты**
	2. *bytearray* – массивы байт
	3. *memoryview* – специальные объекты для доступа к внутренним данным объекта через protocol buffer
7. Множества (*Set Types*)
	1. ***set* – множество**
	2. *frozenset* – неизменяемое множество
8. Словари (*Mapping Types*)
	1. ***dict* – словарь**

**Сложный** (составной, композитный) **тип** — [тип данных](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), объекты (переменные или постоянные) которого имеют внутреннюю структуру, доступную программисту.

В разных [языках программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) набор базовых сложных типов может несколько отличаться (чаще по названию и деталям реализации). Есть, однако, объективные критерии — однотипность элементов и способ доступа, позволяющие выделить главные представители сложных типов.

Подробнее о типизации в языках программирования:

[Ликбез по типизации в языках программирования](https://habr.com/ru/post/161205/)

<https://habr.com/ru/post/161205/>

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

Какие стандартные типы в Python – составные?

Это списки, словари, кортежи, множества.

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) is complex number? **True**

# Список

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]

# Кортеж

Также, 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**.

# Самостоятельная работа

Задачи на операции с составными типами данных:

1. Создать список из целых чисел от 5 до 26.

Это можно сделать в цикле for, при помощи функции range().

Для начала надо создать список. Назовем его «копилка»:

kopilka = []

У списка есть метод append(), который добавляет элемент в список:

>>> kopilka.append(5)

>>> kopilka

[5]

Теперь – в цикле for:

**>>> kopilka = []**

**>>> for i in range(5, 27):**

**... kopilka.append(i)**

**>>> kopilka**

**[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]**

**Не забывайте про обязательный отступ для обозначения тела цикла! Отступ всегда ставится в следующей после двоеточия строке.**

1. Создать список из четных чисел от 6 до 36.

Подсказка: функция range() имеет необязательный аргумент, который говорит, с каким шагом генерировать числа:

**range(6, 37, 2)**

1. Создать список из дробных чисел от -4 до 27 с шагом 0.5
2. Создать список из всех английских букв

То есть, получить список

['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

У каждой буквы есть номер в таблице символов, от 97 до 122.

Чтобы получить из номера символа букву, следует использовать функцию chr():

>>> chr(97)

'a'

1. Объединить списки в один список:

a = [10, 20]

b = [30, 40]

c = [50, 60]

Следует использовать операцию сложения или метод append().

1. Получить строку из списка символов:

Входные данные: a = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

Выходные данные: 'abcdefghijklmnopqrstuvwxyz'

Это можно сделать в цикле, складывая по очереди копилку символов с каждым символом

Альтернатива: применить метод строк join():

>>> "".join(["a"])

'a'

1. Попробовать выполнить задания, заменив список на кортеж. Когда замена работает, а когда – нет?