**Я** начал изучать Python из за того что это высокоуровневый язык программирования общего назначения. Синтаксис языка минималистичен: это нужно, чтобы увеличить производительность разработчика и читаемость кода. **Python** — один из самых популярных и адаптивных языков, который очень быстро развивается. Разработчики на **Python** всегда востребованы на рынке.

**Python** — [высокоуровневый язык программирования](https://ru.m.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%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) общего назначения с динамической типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным — всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. [Синтаксис](https://ru.m.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_(%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%D0%B5)) ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации, сам же язык известен как интерпретируемый и используется в том числе для написания скриптов[]](https://ru.m.wikipedia.org/wiki/Python#cite_note-:2-13). Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на [Си](https://ru.m.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%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)) или [C++](https://ru.m.wikipedia.org/wiki/C%2B%2B).

Python является [мультипарадигмальным языком программирования](https://ru.m.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BF%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%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), поддерживающим [императивное](https://ru.m.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%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%D0%B5), [процедурное](https://ru.m.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%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%D0%B5), [структурное](https://ru.m.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%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%D0%B5), [объектно-ориентированное](https://ru.m.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%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%D0%B5)программирование, [метапрограммирование](https://ru.m.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%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%D0%B5) и [функциональное](https://ru.m.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%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%D0%B5)программирование. Задачи [обобщённого программирования](https://ru.m.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%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%D0%B5) решаются за счёт динамической типизации. [Аспектно-ориентированное программирование](https://ru.m.wikipedia.org/wiki/%D0%90%D1%81%D0%BF%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%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%D0%B5) частично поддерживается через декораторы, более полноценная поддержка обеспечивается дополнительными [фреймворками](https://ru.m.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA). Такие методики как [контрактное](https://ru.m.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D0%BE%D0%B5_%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%D0%B5) и [логическое программирование](https://ru.m.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%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%D0%B5) можно реализовать с помощью библиотек или расширений. Основные архитектурные черты — [динамическая типизация](https://ru.m.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [автоматическое управление памятью](https://ru.m.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0_(%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%D0%B5)), [полная интроспекция](https://ru.m.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_%D0%B2_Python), механизм [обработки исключений](https://ru.m.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), поддержка [многопоточных вычислений](https://ru.m.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C) с глобальной блокировкой интерпретатора ([GIL](https://ru.m.wikipedia.org/wiki/Global_Interpreter_Lock)), высокоуровневые [структуры данных](https://ru.m.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Поддерживается разбиение программ на [модули](https://ru.m.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%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%D0%B5)), которые, в свою очередь, могут объединяться в пакеты.

Эталонной реализацией Python является интерпретатор [CPython](https://ru.m.wikipedia.org/wiki/CPython), поддерживающий большинство активно используемых платформ и являющийся стандартом [де-факто](https://ru.m.wikipedia.org/wiki/%D0%94%D0%B5-%D1%84%D0%B0%D0%BA%D1%82%D0%BE)языка. Он распространяется под [свободной лицензией](https://ru.m.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [Python Software Foundation License](https://ru.m.wikipedia.org/wiki/Python_Software_Foundation_License), позволяющей использовать его без ограничений в любых приложениях, включая [проприетарные](https://ru.m.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D0%B5%D1%82%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E). CPython компилирует исходные тексты в высокоуровневый байткод, который исполняется в стековой виртуальной машине. К другим трём основным реализациям языка относятся [Jython](https://ru.m.wikipedia.org/wiki/Jython) (для [JVM](https://ru.m.wikipedia.org/wiki/JVM)), [IronPython](https://ru.m.wikipedia.org/wiki/IronPython) (для [CLR](https://ru.m.wikipedia.org/wiki/CLR)/[.NET](https://ru.m.wikipedia.org/wiki/.NET_Framework)) и [PyPy](https://ru.m.wikipedia.org/wiki/PyPy). PyPy написан на подмножестве языка Python (RPython) и разрабатывался как альтернатива CPython с целью повышения скорости исполнения программ, в том числе за счёт использования [JIT-компиляции](https://ru.m.wikipedia.org/wiki/JIT-%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F)[[28]](https://ru.m.wikipedia.org/wiki/Python#cite_note-:3-28). Поддержка версии Python 2 закончилась в 2020 году. На текущий момент активно развивается версия языка Python 3. Разработка языка ведётся через предложения по расширению языка **PEP**([англ.](https://ru.m.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Python Enhancement Proposal*), в которых описываются нововведения, делаются корректировки согласно обратной связи от сообщества и документируются итоговые решения[[31]](https://ru.m.wikipedia.org/wiki/Python#cite_note-:7-31).

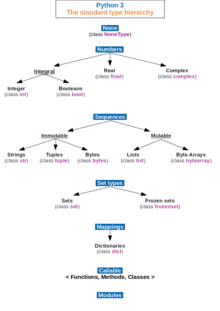
[Стандартная библиотека](https://ru.m.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_Python) включает большой набор полезных переносимых функций, начиная от функционала для работы с текстом и заканчивая средствами для написания сетевых приложений. Дополнительные возможности, такие как математическое моделирование, работа с оборудованием, написание веб-приложений или разработка игр, могут реализовываться посредством обширного количества сторонних библиотек, а также интеграцией библиотек, написанных на Си или C++, при этом и сам интерпретатор Python может интегрироваться в проекты, написанные на этих языках. Существует и специализированный [репозиторий](https://ru.m.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D0%B9)программного обеспечения, написанного на Python, — [PyPI](https://ru.m.wikipedia.org/wiki/PyPI" \o "PyPI). Данный репозиторий предоставляет средства для простой установки пакетов в операционную систему и стал [стандартом де-факто](https://ru.m.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82_%D0%B4%D0%B5-%D1%84%D0%B0%D0%BA%D1%82%D0%BE) для Python. По состоянию на 2019 год в нём содержалось более 175 тысяч пакетов.

Python стал одним из самых популярных языков, он используется в анализе данных, [машинном обучении](https://ru.m.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), [DevOps](https://ru.m.wikipedia.org/wiki/DevOps) и [веб-разработке](https://ru.m.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0), а также в других сферах, включая разработку игр. За счёт читабельности, простого синтаксиса и отсутствия необходимости в компиляции язык хорошо подходит для обучения программированию, позволяя концентрироваться на изучении алгоритмов, концептов и парадигм. Отладка же в значительной степени облегчается тем фактом, что язык является интерпретируемым. Применяется язык многими крупными компаниями, такими как [Google](https://ru.m.wikipedia.org/wiki/Google_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)) или [Facebook](https://ru.m.wikipedia.org/wiki/Facebook)[[13]](https://ru.m.wikipedia.org/wiki/Python#cite_note-:2-13). По состоянию на февраль 2021 года Python занимает третье место в [рейтинге TIOBE](https://ru.m.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81_TIOBE)популярности языков программирования с показателем 10,86 %. «Языком года» по версии TIOBE Python объявлялся в 2007, 2010, 2018 и 2020 году.

**Теперь пройдемся по теории Python**

**Что бы углубиться более точно в этот язык программирования**

**Типы и структуры данных**

[](https://ru.m.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:Python_3._The_standard_type_hierarchy.png)

Python поддерживает [динамическую типизацию](https://ru.m.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), то есть тип переменной определяется только во время исполнения. Поэтому вместо «присваивания значения переменной» лучше говорить о «связывании значения с некоторым именем». В Python имеются встроенные типы: [булевый](https://ru.m.wikipedia.org/wiki/%D0%91%D1%83%D0%BB%D0%B5%D0%B2%D1%81%D0%BA%D0%B8%D0%B9_%D1%82%D0%B8%D0%BF), [строка](https://ru.m.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%BE%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF), [Unicode](https://ru.m.wikipedia.org/wiki/Unicode)-строка, целое число произвольной точности, число [с плавающей запятой](https://ru.m.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B2%D0%B0%D1%8E%D1%89%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D1%8F%D1%82%D0%B0%D1%8F), [комплексное число](https://ru.m.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BB%D0%B5%D0%BA%D1%81%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) и некоторые другие. Из [коллекций](https://ru.m.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%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%D0%B5)) в Python встроены: [список](https://ru.m.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [кортеж](https://ru.m.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D1%82%D0%B5%D0%B6_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))(*неизменяемый список*), [словарь](https://ru.m.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2), [множество](https://ru.m.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)) и другие[[62]](https://ru.m.wikipedia.org/wiki/Python#cite_note-62). Все значения являются объектами, в том числе функции, методы, модули, классы.

Добавить новый тип можно либо написав [класс](https://ru.m.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%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%D0%B5)) (class), либо определив новый тип в модуле расширения (например, написанном на языке C). Система классов поддерживает [наследование](https://ru.m.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%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%D0%B5)) (одиночное и [множественное](https://ru.m.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и [метапрограммирование](https://ru.m.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%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%D0%B5). Возможно наследование от большинства встроенных типов и типов расширений.

Все объекты делятся на изменяемые и неизменяемые: списки, словари и множества являются изменяемыми, а все остальные — неизменяемыми (например, при изменении строки фактически создаётся новая, а при изменении списка — только меняются ссылки в нём). Кортеж в Python является, по сути, неизменяемым списком. Во многих случаях кортежи работают быстрее списков, поэтому если вы не планируете изменять последовательность, то лучше использовать именно их. Неизменяемые объекты (и все объекты в них, если это, например, кортеж) могут быть ключами словаря (должны иметь метод hash).

|  |  |  |  |
| --- | --- | --- | --- |
| **Типы, используемые в Python** | | | |
| **Тип** | [**Изменяемость**](https://ru.m.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82) | **Описание** | **Примеры** |
| bool | Изменяемый | [Логический тип](https://ru.m.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%82%D0%B8%D0%BF) | True False |
| bytearray | Изменяемый | Массив байтов | bytearray(b'Some ASCII') bytearray(b"Some ASCII") bytearray([119, 105, 107, 105]) |
| bytes | Неизменяемый | Массив байтов | b'Some ASCII' b"Some ASCII" bytes([119, 105, 107, 105]) |
| complex | Неизменяемый | [Комплексное число](https://ru.m.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BB%D0%B5%D0%BA%D1%81%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) | 3+2.7j |
| dict | Изменяемый | Словарь ([ассоциативный массив](https://ru.m.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2)), представляет собой коллекцию пар «ключ—значение»; значение может быть любого типа, ключ должен иметь хешируемый тип | {'key1': 1.0, 3: False}  {} |
| ellipsis[[К 1]](https://ru.m.wikipedia.org/wiki/Python" \l "cite_note-%D0%9F-63) | Неизменяемый | [Многоточие](https://ru.m.wikipedia.org/w/index.php?title=%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D1%82%D0%BE%D1%87%D0%B8%D0%B5_(%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%D0%B5)&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Ellipsis_(computer_programming))(элипсис). Используется в основном в [NumPy](https://ru.m.wikipedia.org/wiki/NumPy)для сокращённого задания среза многомерного массива. В самом Python присутствует для поддержки пользовательских типов и таких расширений, как NumPy[[63]](https://ru.m.wikipedia.org/wiki/Python#cite_note-_c8966e1f9acc8d77-64) | ... Ellipsis Для [NumPy](https://ru.m.wikipedia.org/wiki/NumPy):  x[i, ..., j], что эквивалентно x[i, :, :, j][[63]](https://ru.m.wikipedia.org/wiki/Python#cite_note-_c8966e1f9acc8d77-64) |
| float | Неизменяемый | [Число с плавающей запятой](https://ru.m.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%BE_%D1%81_%D0%BF%D0%BB%D0%B0%D0%B2%D0%B0%D1%8E%D1%89%D0%B5%D0%B9_%D0%B7%D0%B0%D0%BF%D1%8F%D1%82%D0%BE%D0%B9). Степень точности зависит от платформы, но на практике обычно реализуется в виде 64-битного 53-разрядного числа[[64]](https://ru.m.wikipedia.org/wiki/Python#cite_note-65) | 1.414 |
| frozenset | Неизменяемый | Неупорядочное [множество](https://ru.m.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)), не содержит дубликатов; может содержать внутри себя различные хешируемые типы данных | frozenset([4.0, 'string', True]) |
| int | Неизменяемый | [Целое число](https://ru.m.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85))неограниченного размера[[65]](https://ru.m.wikipedia.org/wiki/Python#cite_note-pep0237-66) | 42 |
| list | Изменяемый | [Список](https://ru.m.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), может содержать внутри себя различные типы данных | [4.0, 'string', True] [] |
| NoneType[[К 1]](https://ru.m.wikipedia.org/wiki/Python" \l "cite_note-%D0%9F-63) | Неизменяемый | Объект, представляющий собой отсутствие значения, часто называемый [Null](https://ru.m.wikipedia.org/w/index.php?title=Null_(%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%D0%B5)&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Null_pointer)в других языках. | None |
| NotImplementedType[[К 1]](https://ru.m.wikipedia.org/wiki/Python#cite_note-%D0%9F-63) | Неизменяемый | Объект, который возвращяется при [перегрузке операторов](https://ru.m.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BE%D0%B2), когда типы операндов не поддерживаются. | NotImplemented |
| range | Неизменяемый | Последовательность целых чисел от какого-то одного значения до другого, обычно используется для повторения операции несколько раз при помощи for[[66]](https://ru.m.wikipedia.org/wiki/Python" \l "cite_note-67) | range(1, 10) range(10, -5, -2) |
| set | Изменяемый | Неупорядочное [множество](https://ru.m.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)), не содержит дубликатов; может содержать внутри себя различные хешируемые типы данных | {4.0, 'string', True}  set() |
| str | Неизменяемый | [Строковый тип](https://ru.m.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%BE%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF) | 'Wikipedia' "Wikipedia"  *"""Spanning*  *multiple*  *lines"""* |
| tuple | Неизменяемый | [Кортеж](https://ru.m.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D1%82%D0%B5%D0%B6_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)). Может содержать внутри себя различные типы данных. Может использоваться в качестве неизменяемого списка и в качестве записей с неименованными полями[[67]](https://ru.m.wikipedia.org/wiki/Python#cite_note-_431583ef41e1cbf2-68) | В качестве неизменяемого списка:  (4.0, 'string', True) ('single element',) () В качестве записей: lax\_coordinates = (33.9425, -118.408056)  city, year, pop, chg, area = ('Tokyo', 2003, 32450, 0.66, 8014)[[67]](https://ru.m.wikipedia.org/wiki/Python#cite_note-_431583ef41e1cbf2-68) |

Основные свойства

Python не требует явного объявления переменных, является регистро-зависим (переменная var не эквивалентна переменной Var или VAR — это три разные переменные) [объектно-ориентированным](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%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%D0%B5) языком.

Синтаксис

Во первых стоит отметить интересную особенность Python. Он не содержит операторных скобок (begin..end в pascal или {..}в Си), вместо этого **блоки выделяются отступами**: пробелами или табуляцией, а вход в блок из операторов осуществляется двоеточием. Однострочные комментарии начинаются со знака фунта «#», многострочные — начинаются и заканчиваются тремя двойными кавычками «"""».

Чтобы присвоить значение пременной используется знак «=», а для сравнения —   
«==». Для увеличения значения переменной, или добавления к строке используется оператор «+=», а для уменьшения — «-=». Все эти операции могут взаимодействовать с большинством типов, в том числе со строками. Например

>>> myvar = 3  
>>> myvar += 2  
>>> myvar -= 1  
""«Это многострочный комментарий   
Строки заключенные в три двойные кавычки игнорируются»""  
>>> mystring = «Hello»  
>>> mystring += " world."  
>>> **print** mystring  
Hello world.  
# Следующая строка меняет   
значения переменных местами. (Всего одна строка!)  
>>> myvar, mystring = mystring, myvar

Структуры данных

Python содержит такие структуры данных как**списки (lists), кортежи (tuples) и словари (dictionaries**). Списки — похожи на одномерные массивы (но вы можете использовать Список включающий списки — многомерный массив), кортежи — неизменяемые списки, словари — тоже списки, но индексы могут быть любого типа, а не только числовыми. "Массивы" в Python могут содержать данные любого типа, то есть в одном массиве может могут находиться числовые, строковые и другие типы данных. Массивы начинаются с индекса 0, а последний элемент можно получить по индексу -1 Вы можете присваивать переменным функции и использовать их соответственно.

>>> sample = [1, [«another», «list»], («a», «tuple»)] #Список состоит из целого числа, другого списка и кортежа  
>>> mylist = [«List item 1», 2, 3.14] #Этот список содержит строку, целое и дробное число  
>>> mylist[0] = «List item 1 again» #Изменяем первый (нулевой) элемент листа mylist  
>>> mylist[-1] = 3.14 #Изменяем последний элемент листа  
>>> mydict = {«Key 1»: «Value 1», 2: 3, «pi»: 3.14} #Создаем словарь, с числовыми и целочисленным индексами  
>>> mydict[«pi»] = 3.15 #Изменяем элемент словаря под индексом «pi».  
>>> mytuple = (1, 2, 3) #Задаем кортеж  
>>> myfunction = len #Python позволяет таким образом объявлять синонимы функции  
>>> **print** myfunction(list)   
3

Вы можете использовать часть массива, задавая первый и последний индекс через двоеточие «:». В таком случае вы получите часть массива, от первого индекса до второго не включительно. Если не указан первый элемент, то отсчет начинается с начала массива, а если не указан последний — то масив считывается до последнего элемента. Отрицательные значения определяют положение элемента с конца. Например:

>>> mylist = [«List item 1», 2, 3.14]  
>>> **print** mylist[:] #Считываются все элементы массива  
['List item 1', 2, 3.1400000000000001]  
>>> **print** mylist[0:2] #Считываются нулевой и первый элемент массива.  
['List item 1', 2]  
>>> **print** mylist[-3:-1] #Считываются элементы от нулевого (-3) до второго (-1) (не включительно)  
['List item 1', 2]  
>>> **print** mylist[1:] #Считываются элементы от первого, до последнего  
[2, 3.14]

Строки

Строки в Python **обособляются кавычками двойными «"» или одинарными «'»**. Внутри двойных ковычек могут присутствовать одинарные или наоборот. К примеру строка «Он сказал 'привет'!» будет выведена на экран как «Он сказал 'привет'!». Если нужно использовать строку из несколько строчек, то эту строку надо начинать и заканчивать тремя двойными кавычками «"""». Вы можете подставить в шаблон строки элементы из кортежа или словаря. Знак процента «%» между строкой и кортежем, заменяет в строке символы «%s» на элемент кортежа. Словари позволяют вставлять в строку элемент под заданным индексом. Для этого надо использовать в строке конструкцию «%(индекс)s». В этом случае вместо «%(индекс)s» будет подставлено значение словаря под заданным индексом.

>>>**print** «Name: %s\nNumber: %s\nString: %s» % (my**class**.name, 3, 3 \* "-")  
Name: Poromenos  
Number: 3  
String: —    
strString = ""«Этот текст расположен  
на нескольких строках»""  
   
>>> **print** «This %(verb)s a %(noun)s.» % {«noun»: «test», «verb»: «is»}  
This is a test.

Операторы

Операторы while, **if**, **for** составляют операторы перемещения. Здесь нет аналога оператора select, так что придется обходиться **if**. В операторе **for** происходит сравнение **переменной и списка**. Чтобы получить список цифр до числа <number> — используйте функцию range(<number>). Вот пример использования операторов

rangelist = range(10) #Получаем список из десяти цифр (от 0 до 9)  
>>> **print** rangelist  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
**for** number in rangelist: #Пока переменная number (которая каждый раз увеличивается на единицу) входит в список…   
# Проверяем входит ли переменная  
# numbers в кортеж чисел (3, 4, 7, 9)  
**if** number in (3, 4, 7, 9): #Если переменная number входит в кортеж (3, 4, 7, 9)...  
# Операция «**break**» обеспечивает  
# выход из цикла в любой момент  
**break**   
**else**:  
# «**continue**» осуществляет «прокрутку»  
# цикла. Здесь это не требуется, так как после этой операции   
# в любом случае программа переходит опять к обработке цикла  
**continue**  
**else**:  
# «**else**» указывать необязательно. Условие выполняется

# если цикл не был прерван при помощи «**break**».  
**pass** # Ничего не делать  
   
**if** rangelist[1] == 2:  
**print** «The second item (lists are 0-based) is 2»  
**elif** rangelist[1] == 3:  
**print** «The second item (lists are 0-based) is 3»  
**else**:  
**print** «Dunno»  
   
**while** rangelist[1] == 1:  
**pass**

Функции

Для объявления функции служит **ключевое слово «def»**. Аргументы функции задаются в скобках после названия функции. Можно задавать необязательные аргументы, присваивая им значение по умолчанию. Функции могут возвращать кортежи, в таком случае надо писать возвращаемые значения через запятую. Ключевое слово «**lambda**» служит для объявления элементарных функций .

# arg2 и arg3 — необязательые аргументы, принимают значение объявленное по умолчни,  
# если не задать им другое значение при вызове функци.  
**def** myfunction(arg1, arg2 = 100, arg3 = «test»):  
**return** arg3, arg2, arg1  
#Функция вызывается со значением первого аргумента — "Argument 1", второго — по умолчанию, и третьего — "Named argument".  
>>>ret1, ret2, ret3 = myfunction(«Argument 1», arg3 = «Named argument»)  
# ret1, ret2 и ret3 принимают значения "Named argument", 100, "Argument 1" соответственно  
>>> **print** ret1, ret2, ret3  
Named argument 100 Argument 1  
   
# Следующая запись эквивалентна **def** f(x): **return** x + 1  
functionvar = **lambda** x: x + 1  
>>> **print** functionvar(1)  
2

Классы

Язык Python ограничен в множественном наследовании в классах. Внутренние переменные и внутренние методы классов начинаются с двух знаков нижнего подчеркивания «\_\_» (например «\_\_myprivatevar»). Мы можем также присвоить значение переменной класса извне. Пример:

**class** My**class**:  
common = 10  
**def** \_\_init\_\_(self):  
self.myvariable = 3  
**def** myfunction(self, arg1, arg2):  
**return** self.myvariable  
   
# Здесь мы объявили класс My**class**. Функция \_\_init\_\_ вызывается автоматически при инициализации классов.  
>>> classinstance = My**class**() # Мы инициализировали класс и переменная myvariable приобрела значение 3 как заявлено в методе инициализации  
>>> classinstance.myfunction(1, 2) #Метод myfunction класса My**class** возвращает значение переменной myvariable  
3  
# Переменная common объявлена во всех классах  
>>> classinstance2 = My**class**()  
>>> classinstance.common  
10  
>>> classinstance2.common  
10  
# Поэтому, если мы изменим ее значение в классе My**class** изменятся  
# и ее значения в объектах, инициализированных классом My**class**   
>>> Myclass.common = 30  
>>> classinstance.common  
30  
>>> classinstance2.common  
30  
# А здесь мы не изменяем переменную класса. Вместо этого  
# мы объявляем оную в объекте и присваиваем ей новое значение   
>>> classinstance.common = 10  
>>> classinstance.common  
10  
>>> classinstance2.common  
30  
>>> Myclass.common = 50  
# Теперь изменение переменной класса не коснется   
# переменных объектов этого класса  
>>> classinstance.common  
10  
>>> classinstance2.common  
50  
   
# Следующий класс является наследником класса My**class**  
# наследуя его свойства и методы, ктому же класс может   
# наследоваться из нескольких классов, в этом случае запись   
# такая: **class** Otherclass(Myclass1, Myclass2, MyclassN)  
**class** Otherclass(Myclass):  
**def** \_\_init\_\_(self, arg1):  
self.myvariable = 3  
**print** arg1  
   
>>> classinstance = Otherclass(«hello»)  
hello  
>>> classinstance.myfunction(1, 2)  
3  
# Этот класс не имеет совйтсва test, но мы можем   
# объявить такую переменную для объекта. Причем  
# tэта переменная будет членом только **class**instance.  
>>> classinstance.test = 10  
>>> classinstance.test  
10

Исключения

Исключения в Python имеют структуру **try**-**except** [**except**ionname]:

**def** somefunction():  
**try**:  
# Деление на ноль вызывает ошибку  
10 / 0  
**except** ZeroDivisionError:  
# Но программа не "Выполняет недопустимую операцию"  
# А обрабатывает блок исключения соответствующий ошибке «ZeroDivisionError»  
**print** «Oops, invalid.»  
   
>>> fn**except**()  
Oops, invalid.

Импорт

Внешние библиотеки можно подключить процедурой «**import** [libname]», где [libname] — название подключаемой библиотеки. Вы так же можете использовать команду «**from** [libname] **import** [funcname]», чтобы вы могли использовать функцию [funcname] из библиотеки [libname]

**import** random #Импортируем библиотеку «random»  
**from** time **import** clock #И заодно функцию «clock» из библиотеки «time»  
   
randomint = random.randint(1, 100)  
>>> **print** randomint  
64

Работа с файловой системой

Python имеет много встроенных библиотек. В этом примере мы попробуем сохранить в бинарном файле структуру списка, прочитать ее и сохраним строку в текстовом файле. Для преобразования структуры данных мы будем использовать стандартную библиотеку «pickle»

**import** pickle  
mylist = [«This», «is», 4, 13327]  
# Откроем файл C:\binary.dat для записи. Символ «r»   
# предотвращает замену специальных сиволов (таких как \n, \t, \b и др.).  
myfile = file(r«C:\binary.dat», «w»)  
pickle.dump(mylist, myfile)  
myfile.close()  
   
myfile = file(r«C:\text.txt», «w»)  
myfile.write(«This is a sample string»)  
myfile.close()  
   
myfile = file(r«C:\text.txt»)  
>>> **print** myfile.read()  
'This is a sample string'  
myfile.close()  
   
# Открываем файл для чтения  
myfile = file(r«C:\binary.dat»)  
loadedlist = pickle.load(myfile)  
myfile.close()  
>>> **print** loadedlist  
['This', 'is', 4, 13327]

Особенности

* **Условия могут комбинироваться.** 1 < a < 3 выполняется тогда, когда а больше 1, но меньше 3.
* Используйте операцию «**del**» чтобы **очищать переменные или элементы массива**.
* Python предлагает большие возможности для **работы со списками**. Вы можете использовать операторы объявлении структуры списка. Оператор **for** позволяет задавать элементы списка в определенной последовательности, а **if** — позволяет выбирать элементы по условию.

>>> lst1 = [1, 2, 3]  
>>> lst2 = [3, 4, 5]

>>> **print** [x \* y **for** x in lst1 **for** y in lst2]  
[3, 4, 5, 6, 8, 10, 9, 12, 15]  
>>> **print** [x **for** x in lst1 **if** 4 > x > 1]  
[2, 3]  
# Оператор «any» возвращает true, если хотя   
# бы одно из условий, входящих в него, выполняется.  
>>> any(i % 3 **for** i in [3, 3, 4, 4, 3])  
True  
# Следующая процедура подсчитывает количество   
# подходящих элементов в списке  
>>> sum(1 **for** i in [3, 3, 4, 4, 3] **if** i == 3)  
3  
>>> **del** lst1[0]  
>>> **print** lst1  
[2, 3]  
>>> **del** lst1

* **Глобальные переменные**объявляются вне функций и могут быть прочитанны без каких либо объявлений. Но если вам необходимо изменить значение глобальной переменной из функции, то вам необходимо объявить ее в начале функции ключевым словом «**global**», если вы этого не сделаете, то Python объявит переменную, доступную только для этой функции.

number = 5  
   
**def** myfunc():  
# Выводит 5  
**print** number  
   
**def** anotherfunc():  
# Это вызывает исключение, поскольку глобальная апеременная   
# не была вызванна из функции. Python в этом случае создает   
# одноименную переменную внутри этой функции и доступную  
# только для операторов этой функции.  
**print** number  
number = 3  
   
**def** yetanotherfunc():  
**global** number  
# И только из этой функции значение переменной изменяется.  
number = 3.

Преимущества Python

* Скорость выполнения программ написанных на Python очень высока. Это связанно с тем, что основные библиотеки Python   
  написаны на C++ и выполнение задач занимает меньше времени, чем на других языках высокого уровня.
* В связи с этим вы можете писать свои собственные модули для Python на C или C++
* В стандартныx библиотеках Python вы можете найти средства для работы с электронной почтой, протоколами   
  Интернета, FTP, HTTP, базами данных, и пр.
* Скрипты, написанные при помощи Python выполняются на большинстве современных ОС. Такая переносимость обеспечивает Python применение в самых различных областях.
* Python подходит для любых решений в области программирования, будь то офисные программы, вэб-приложения, GUI-приложения и т.д.
* Над разработкой Python трудились тысячи энтузиастов со всего мира. Поддержкой современных технологий в стандартных библиотеках мы можем быть обязаны именно тому, что Python был открыт для всех желающих.

**ВЫВОД:**

Мы изучили множество вещей о базах Python:

* **Типы структуры данных**
* **Основные свойства**
* **Синтаксис**
* **Структуры данных**
* **Строки**
* **Операторы**
* **Функций**
* **Классы**
* **Исключения**
* **Импорт**
* **Работа с файловой системой**
* **Особенности**

**ЛИТЕРАТУРА:**

* 1. <https://ru.m.wikipedia.org/wiki/Python>
  2. https://www.google.com.ua/amp/s/m.habr.com/ru/amp/post/31180/