**Почему Python?)**

Для многих основное преимущество языка Python заключается в удобочитаемости, ясности и, главное, Python является [востребованным](https://dou.ua/lenta/articles/language-rating-jan-2019/) языком программирования, на котором можно написать практически все. Например, создавать веб-проекты, заниматься машинным обучением и анализом данных, автоматизировать задачи системного администрирования, создавать игры и многое другое.

Большая часть программ на языке Python выполняется без изменений на всех основных платформах. Перенос программного кода из операционной системы Linux в Windows обычно заключается в простом копировании файлов программ с одной машины на другую.

Богатая стандартная библиотека является одной из привлекательных сторон языка программирования Python. Эта библиотека предоставляет массу возможностей, востребованных в прикладных программах, начиная от поиска текста по шаблону и заканчивая сетевыми функциями. Кроме того, Python допускает расширение и за счет сторонних библиотек.

Тут же стоит отметить большое дружелюбное сообщество Python. Существует множество курсов, туториалов, и просто интересных статей, так что разработчик сможет найти для себя что-то интересное. По [данной ссылке](https://docs.python-guide.org/intro/learning/) вы можете найти различные англоязычные ресурсы для обучения Python.

**О языке**

Разработка языка [Python](https://ru.wikipedia.org/wiki/Python#cite_note-13) была начата в конце 1980-х годов сотрудником голландского института CWI Гвидо ван Россумом. В феврале 1991 года Гвидо опубликовал исходный код в группе новостей alt.sources. Название язык получил в честь популярного телешоу «Летающий цирк Монти Пайтона».

Python – интерпретируемый язык. Это значит, что выполнение код происходит инструкция за инструкцией.

Python изначально является объектно-ориентированным языком программирования, однако поддерживает ии другие парадигмы программирования такие как: структурное, функциональное, императивное и аспектно-ориентированное. При наличии соответствующего связующего программного кода, программы на языке Python могут использовать механизм наследования от классов, реализованных на C++, Java и C#.

Важным событием в истории Python был момент, когда разработчики выпустили третью версию, обратно несовместимую со второй, чтобы решить некоторые архитектурные недостатки второй версии языка. Обратная несовместимость привела к тому, что до сих пор многие продакшн-системы используют Python версии 2. Тем не менее, в 2020-м году официальная поддержка Python 2 прекратится, поэтому новые проекты стоит начинать именно на Python 3.

Стоит отметить, что Python – это название спецификации языка. Есть основные три реализации языка: CPython, Jython и IronPython. CPython – это стандартная реализация, а все остальные создавались для специфических целей и задач. Jython – это альтернативная реализация языка Python, основная цель которой – тесная интеграция с языком программирования Java, а IronPython –интеграция с приложениями, созданными для работы в среде Microsoft .NET Framework, а также в Mono – открытом эквиваленте для операционной системы Linux. Все три реализации, CPython, Jython и IronPython, реализуют язык Python похожими способами: исходный программный код компилируют в байт-код и выполняют полученный байт-код с помощью соответствующей виртуальной машины.

Есть и другие реализации спецификации языка, например, Проект PyPy предлагает реализацию Python с использованием JIT-компиляции, которая значительно увеличивает скорость выполнения Python-программ.

**Что можно делать с помощью Python?**

1. **Системное программирование.** Встроенные в Python интерфейсы доступа к службам операционных систем  
   делают его идеальным инструментом для создания переносимых программ и утилит системного администрирования (иногда они называются *инструментами  командной  оболочки*). Программы на языке Python могут отыскивать файлы и каталоги, запускать другие программы, производить параллельные вычисления с использованием нескольких процессов и потоков и делать многое другое.
2. **Графический интерфейс.** Простота Python и высокая скорость разработки делают его отличным средством разработки графического интерфейса. состав Python входит стандартный объектно-ориентированный интерфейс к Tk GUI API, который называется tkinter, позволяющий программам на языке Python реализовать переносимый графический интерфейс с внешним видом,

присущим операционной системе. При выборе [соответствующей библиотеки](https://docs.python.org/3/library/othergui.html#other-gui-packages) также можно использовать другие инструменты создания  
графического интерфейса, такие как:

* *Qt* (с помощью PyQt);
* *GTK* (с помощью PyGtk);
* *MFC* (с помощью PyWin32);
* *.NET* (с помощью IronPython);
* *Swing* (с помощью Jython – реализации языка Python на Java).

Для разработки приложений с веб-интерфейсом или не предъявляющих высоких требований к интерфейсу можно использовать Jython, веб-фреймворки на языке Python и CGI-сценарии.

1. **Веб-сценарии.** Интерпретатор Python поставляется вместе со стандартными интернет-модулями, которые позволяют программам выполнять разнообразные сетевые операции как в режиме клиента, так и в режиме сервера. Кроме того, существует огромная коллекция сторонних инструментов для создания сетевых программ на языке Python, которые можно найти в Интернете.
2. **Приложения баз данных.** В языке Python имеются интерфейсы доступа ко всем основным реляционным базам данных – Sybase, Oracle, Informix, ODBC, MySQL, PostgreSQL, SQLite и многим другим. В мире Python существует также переносимый прикладной   
   программный интерфейс баз данных, предназначенный для доступа к базам данных SQL из сценариев на языке Python, который унифицирует доступ к различным базам данных.
3. **Программирование математических и научных вычислений.** Одним из примеров может бытьрасширение NumPy для математических вычислений, которое включает такие мощные элементы, как объекты массивов, интерфейсы к стандартным математическим библиотекам, и многое другое.
4. **Игры, изображения, искусственный интеллект,  
   XML роботы и многое другое.** Язык программирования Python можно использовать для:

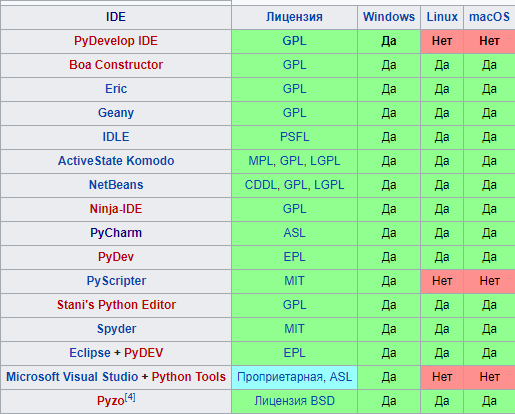
* Создание игровых программы и анимационных роликов с помощью системы *pygame*
* Обмен данными с другими компьютерами через последовательный порт с помощью расширения *PySerial*
* Обработка изображений с помощью расширений *PIL*, *PyOpenGL*, *Blender*, *Maya* и других
* Управление роботом с помощью инструмента *PyRo*
* Разбора XML-документов с помощью пакета xml, модуля xmlrpclib и расширений сторонних разработчиков
* Программировать искусственный интеллект с помощью эмулятора нейросетей и оболочек экспертных систем

Поддержку многих других прикладных областей можно найти на веб-сайте [PyPI](https://pypi.org/).

**Установка Python**

Python – не только язык программирования, но еще и программный пакет, который называется интерпретатором. В процессе установки пакета Python на компьютер создается ряд программных компонентов – как минимум, интерпретатор и библиотека поддержки. В зависимости от особенностей использования интерпретатор Python может иметь вид исполняемой программы или набора библиотек, связанных с другой программой. В зависимости от версии Python сам интерпретатор может быть реализован как программа на языке C, как набор классов Java или в каком-либо другом виде.

Для начала необходимо [установить](https://www.python.org/downloads/) интерпретатор Python. Код можно писать в любом текстовом редакторе. Например, Notepad++ обеспечивает подсветку синтаксиса. Однако, существуют различные IDE, которые добавляют процессу программирования массу удобств – таких как подсветка синтаксиса, автодополнение, интроспекция кода (возможность по клику перейти к месту объявления используемого класса или функции) и многие другие. В следующей таблице представлены различные IDE:



Есть еще дистрибутив Anaconda. Он очень популярен у Data Scientist и Machine Learning Engineer. В него по умолчанию входит Spyder IDE, и Jupiter Notebook. В [этой статье](https://www.quora.com/What-is-the-difference-between-PyCharm-and-Anaconda-for-Python) хорошо описывается в чем отличия между PyCharm и Anaconda. На начальном этапе знакомства с Python можно ограничится работой в IDLE – это интегрированная среда разработки и обучения на языке Python, созданная с помощью библиотеки Tkinter. Официально – искажение IDE, но на самом деле названа в честь Эрика Айдла (англ. Eric Idle) из Монти Пайтон. Поставляется вместе с Python.

После установки Python у вас есть возможность выбора пункта IDLE, который запускает среду разработки с графическим интерфейсом, а пункт Pythonзапускает простой интерактивный сеанс работы с интерпретатором. Кроме того, здесь же можно увидеть пункт вызова стандартного справочного руководства.

Напишем небольшой код. Запустите Python IDLE. В строке напишите следующий код:

print(“print something”)

Таким образом, вы можете писать или проводить тесты прямо в интерактивной оболочке, или можно создать файл. Чтобы создать (или отредактировать) файл с исходным программным кодом в среде IDLE, откройте окно текстового редактора: в главном окне откройте меню File (Файл) и выберите пункт New Window (Новое окно), чтобы открыть окно текстового редактора (или Open… (Открыть) – чтобы отредактировать существующий файл).

Чтобы запустить файл с программным кодом в среде IDLE, выберите окно, где редактируется текст, раскройте меню Run и выберите в нем пункт Run Module. Если с момента открытия или последнего сохранения файла его содержимое изменялось, Python предложит сохранить его.

Когда сценарий запускается таким способом, весь вывод, который он генерирует, а также все сообщения об ошибках появляются в основном окне интерактивного сеанса работы с интерпретатором (командная оболочка Python).

**Красивый код**

Как в любом языке, Python имеет руководство по написанию кода – [Python Enhancement Proposals](https://www.python.org/dev/peps/), или коротко PEP. PEP – это проектный документ, предоставляющий информацию сообществу Python или описывающий новую функцию для Python, его процессов или среды. Данный документ предоставляет краткую техническую спецификацию функции и обоснование этой функции.

Предполагается, что PEP является основным механизмом для предложения новых функций, для сбора отзывов сообщества по проблеме, для документирования проектных решений, которые вошли в Python. Автор PEP отвечает за достижение консенсуса в сообществе и документирование несогласных мнений.

Существует три вида PEP:

1. «A Standards Track» описывает новую функцию или реализацию Python. Он также может описывать стандарт взаимодействия, который будет поддерживаться вне стандартной библиотеки для текущих версий Python до того, как последующий PEP добавит поддержку стандартной библиотеки в будущей версии.
2. Informational PEP – описывает проблему разработки Python или предоставляет общие рекомендации, информацию для сообщества Python, но не предлагает новую функцию. Информационные PEP не обязательно представляют собой рекомендации сообщества Python, поэтому пользователи и разработчики сами решают следовать ли советам информационных PEP.
3. Process PEP – описывает процесс, окружающий Python, или предлагает изменение (или событие в) процесса. Process PEP подобен Standards Track PEPs, но применяется к областям, отличным от самого языка Python. Process PEP могут предлагать реализацию, но не кодовую базу Python; они часто требуют общественного согласия; в отличие от информационных PEP, они являются чем-то большим, чем рекомендации, и пользователи обычно не могут их игнорировать. Примеры включают процедуры, рекомендации, изменения в процессе принятия решений и изменения в инструментах или среде, используемых при разработке Python. Любой мета-PEP также считается процессом PEP.

Начать знакомство можно с [PEP8](https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html) – рекомендации по стилевому оформлению. Рекомендации о стиле написания кода направлены на то, чтобы улучшить читаемость кода и сделать его согласованным между большим числом проектов. В идеале, весь код будет написан в едином стиле, и любой сможет легко его прочесть. Коротко можно почитать [тут](https://geekbrains.ru/posts/pep8).

Еще, при создании переменных важно помнить про зарезервированные имена встроенных функций, имен модулей. [Пример](https://pythonworld.ru/osnovy/vstroennye-funkcii.html)…

**Базовые типы и конструкции**

Данные в языке Python представлены в форме объектов – либо встроенных, предоставляемых языком Python, либо объектов, которые создаются с применением конструкций языка или других инструментов.

Переменные в языке Python, в отличие от других языков программирования, всегда являются указателями на объекты, а не метками областей памяти, доступных для изменения: запись нового значения в переменную не приводит к изменению первоначального объекта, но приводит к тому, что переменная начинает ссылаться на совершенно другой объект. В результате инструкция присваивания может воздействовать только на одну переменную, если мы работаем с неизменяемыми объектами. Однако, когда в уравнении появляются изменяемые объекты и операции, их изменяющие, картина несколько меняется. Это мы рассмотрим при изучении изменяемых объектов.

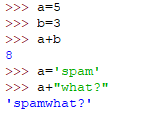
Каждый объект имеет два стандартных поля: указатель на тип, используемый для хранения информации о типе объекта, и счетчик ссылок, используемый для определения момента, когда память, занимаемая объектом, может быть освобождена. Многие объекты расширяют эту структуру, добавляя необходимые поля, но эти два поля должны присутствовать в любом случае.

Счётчик ссылок — это число, показывающее, сколько раз другие объекты ссылаются на данный. В коде a = b = c = object() инициализируется пустой объект и связывается с тремя разными именами: a, b и c. Каждое имя создаёт новую ссылку на объект, но при этом объект создаётся единожды. Связывание объекта с новым именем или добавление объекта в список создаёт новую ссылку, но не создаёт новый объект. Всякий раз, когда имя переменной ассоциируется с новым объектом, интерпретатор Python освобождает память, занимаемую предыдущим объектом (если на него не ссылается какое-либо другое имя или объект). Такое автоматическое освобождение памяти, занимаемой объектами, называется сборкой мусора (garbage collection).

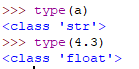
Второе поле – указатель на тип объекта, – является центральным понятием объектной модели Python. У каждого объекта всего один тип, который не меняется в течение жизни объекта и, что важно, тип объекта (и только тип объекта) определяет, что можно с ним делать. Тип также является объектом (у него есть и счётчик ссылок, и его собственный тип; тип большинства типов — type), но в дополнение к двум основным полям он содержит множество других полей. Таким образом, типы в языке Python – это свойства объектов, а не переменных. Больше почитать о внутренней структуре языка можно [тут](https://docs.python.org/3/c-api/typeobj.html#type-objects) или [вот статья](https://habr.com/ru/company/buruki/blog/189986/) (а также глава 7 Изучаем Python O`Reilly).

Python является языком с динамической типизацией. Он определяет тип данных переменной исходя из значения, которое ей присвоено. В данном языке программирования не требуется заранее объявлять переменные, но прежде чем их использовать, им должны быть присвоены некоторые значения.

Объекты могут создаваться путем присваивания литералов переменным, например, х = 17, или обращением к имени соответствующего типа как к функции, например, х = int(17). Некоторые объекты (например, типа decimal. Decimal) могут создаваться только посредством использования их типов, так как они не имеют литерального представления.



С помощью функции type() динамически можно узнать текущий тип переменной:



Все объекты в языке Python либо относятся к классу изменяемых, либо нет. Если говорить о базовых типах, то числа, строки и кортежи являются неизменяемыми, а списки, множества и словари – нет (однако есть и неизменяемые множества☺).

В таблице представлены встроенные (базовые) типы объектов:

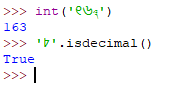
|  |  |
| --- | --- |
| **Тип объекта** | **Пример** |
| Числа | 234, 3.345, 1+2j, Decimal, Fraction |
| Строки | ‘spam’, “guido’s” , b’a\x01c’ |
| Списки | [1, [2, ‘three’], 4] |
| Словари | {‘food’: ‘spam’, ‘taste’: ‘yum’} |
| Кортежи | (1,’spam’, 4, ‘U’) |
| Файлы | myfile = open(‘eggs’, ‘r’) |
| Множества | set(‘abc’), {‘a’, ‘b’, ‘c’} |
| Прочие базовые типы | Сами типы, None, логические значения |
| Типы структурных элементов  программ | Функции, модули, классы |
| Типы, имеющие отношение  к реализации | Компилированный программный код,  стек вызовов |

Далее рассмотрим их детальнее.

**Числа**

Язык Python поддерживает обычные числовые типы (целые и вещественные), а также литералы – для их создания, и выражения – для их обработки. Кроме того, Python предоставляет дополнительную поддержку чисел и объектов для работы с ними.

Python соблюдает правила Unicode и считает цифрами несколько сотен символов. Полный список можно найти [тут](http://www.fileformat.info/info/unicode/category/Nd/list.htm).



Ниже приводится полный перечень числовых типов и инструментов, поддерживаемых в языке Python:

• Целые и вещественные числа

• Комплексные числа

• Числа фиксированной точности

• Рациональные числа

• Множества

• Логические значения

• Целые числа неограниченной точности

• Различные встроенные функции и модули для работы с числами