**Лекция 1.**

**История языка Python.**

Python берет свое начало в 1989 году. Голландский программист Гвидо ван Россум, сотрудник института CWI, занимался созданием языка ABC на ОС Amoeba. Разработка его, видимо, чем-то не устраивала, потому что Россум решил создать свой собственный язык программирования, который был назван в честь передачи «Воздушный цирк Монти Пайтона» - Python.

В 1991 году Python был размещен в домене общего доступа. Распространялся Python бесплатно, не имел никаких ограничений в условиях применений и не требовались авторские отчисления при коммерческом использовании.

Именно эти факторы, в совокупности с талантами Россума позволили языку Python быстро набрать популярность.

Сначала это был просто небольшой интерпретатор с некоторым количеством функций. В течение 1991 года появились первые средства объектно-ориентированного программирования.

Через некоторое время Гвидо переехал из Голландии в Америку. Его пригласили работать в корпорации CNRI. Он там работал, используя Python для тех проектов, которыми занималась корпорация, а в свободное рабочее время развивал язык и интерпретатор.

Так это продолжалось до появления в 1999 году версии 1.5.2. К тому времени в Интернете сложилось немалое сообщество пользователей. Гвидо решил пожертвовать частью зарплаты и стал искать спонсора, который позволил бы ему работать именно над развитием языка и интерпретатора. И образовавшаяся как раз в то время фирма BeOpen решила его поддержать. Он ушел, но перед уходом обязался выпустить версию 1.6. Это была контрактная версия, которую он должен был выпустить в корпорации CNRI. В BeOpen он выпустил версию 2.0.

Версия 2.0 - это довольно большой шаг вперед, но в первую очередь в социальном плане. В этой версии самое главное, что процесс развития языка и интерпретатора стал более открытым. Гвидо перевел все на SourceForge. Процесс развития всех альфа версий стал виден гораздо большему количеству народа. Сообщество пользователей выросло и оно стало требовать возможности участия в развитии языка, и Гвидо на это пошел.

Кроме этой активности произошло много чего нового: появился Юникод; это большой шаг вперед. К Юникоду, соответственно, появился новый механизм регулярных выражений, который работает как для обычных строк, так и для Юникодовых.

Вышел Питон 2.1. В нем появились объекты из языков программирования под названием closures. Правильнее было бы сказать - статически вложенные функции, что следует наверное объяснить в терминах языка Pascal, потому что в языке C такого нет, есть только функции верхнего уровня; в Perl тоже такого нет, хотя вложенные функции есть. В Питоне появилась иерархия: можно функции вкладывать друг в друга, и они будут иметь доступ к переменным окружающих функций, и это большой шаг вперед. Он сильно меняет язык, меняет многие подходы и способы программирования. В версии 2.1 это опционально, но в версии 2.2 это уже будет стандартным свойством языка.

В ноябре 2014 было объявлено, что Python 2.7 будет поддерживаться до 2020 года, и подтверждено, что релиза 2.8 не будет, так как предполагается, что пользователи должны переходить на версию 3+ при первой же возможности.

Python 3.0 (называемый также "Python 3000" или "Py3K") разрабатывался с целью устранения фундаментальных изъянов в языке. Эти изменения не могли быть сделаны при условии сохранения полной обратной совместимости с 2.x версией, поэтому потребовалось изменение главного номера версии. Ведущим принципом разработки Python 3 было: «уменьшение дублирующейся функциональности устранением устаревших способов сделать это».

**Python в наше время.**

**Python** — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода.

Объектно-ориентированная природа Python, являясь мощным средством структурирования программного кода многократного пользования, кроме того, делает этот язык идеальным инструментом поддержки сценариев для объектно-ориентированных языков, таких как C++ и Java. Например, при наличии соответствующего связующего программного кода, программы на языке Python могут использовать механизм наследования от классов, реализованных на C++, Java и С#.

Python может использоваться и распространяться совершенно бесплатно. Как и в случае с другими открытыми программными продуктами, такими как Tel, Perl, Linux и Apache, вы сможете получить в Интернете полные исходные тексты реализации Python. Нет никаких ограничений на его копирование, встраивание в свои системы или распространение в составе ваших продуктов.

Фактически вы сможете даже продавать исходные тексты Python, если появится такое желание.

С точки зрения функциональных возможностей Python можно назвать гибридом. Его инструментальные средства укладываются в диапазон между традиционными языками сценариев (такими как Tel, Scheme и Perl) и языками разработки программных систем (такими как С, C++ и Java). Python обеспечивает простоту и непринужденность языка сценариев и мощь, которую обычно можно найти в компилирующих языках. Превышая возможности других языков сценариев, такая комбинация делает Python удобным средством разработки крупномасштабных проектов. Для предварительного ознакомления ниже приводится список основных возможностей, которые есть в арсенале Python:

**Динамическая типизация**

Python сам следит за типами объектов, используемых в программе, благодаря чему не требуется писать длинные и сложные объявления в программном коде. В действительности, как вы увидите в главе 6, в языке Python вообще отсутствуют понятие типа и необходимость объявления переменных. Так как программный код на языке Python не стеснен рамками типов данных, он автоматически может обрабатывать целый диапазон объектов.

**Автоматическое управление памятью**

Python автоматически распределяет память под объекты и освобождает ее («сборка мусора»), когда объекты становятся ненужными. Большинство объектов могут увеличивать и уменьшать занимаемый объем памяти по мере необходимости. Как вы узнаете, Python сам производит все низкоуровневые операции с памятью, поэтому вам не придется беспокоиться об этом.

**Модульное программирование**

Для создания крупных систем Python предоставляет такие возможности, как модули, классы и исключения. Они позволяют разбить систему на составляющие, применять ООП для создания программного кода многократного пользования и элегантно обрабатывать возникающие события и ошибки.

**Встроенные типы объектов**

Python предоставляет наиболее типичные структуры данных, такие как списки, словари и строки, в виде особенностей, присущих самому языку программирования. Как вы увидите позднее, эти типы отличаются высокой гибкостью и удобством. Например, встроенные объекты могут расширяться и сжиматься по мере необходимости, могут комбинироваться друг с другом для представления данных со сложной структурой и многое другое.

**Встроенные инструменты**

Для работы со всеми этими типами объектов в составе Python имеются мощные и стандартные средства, включая такие операции, как конкатенация (объединение коллекций), получение срезов (извлечение части коллекции), сортировка, отображение и многое другое.

**Библиотеки утилит**

Для выполнения более узких задач в состав Python также входит большая коллекция библиотечных инструментов, которые поддерживают практически все, что только может потребоваться, - от поиска с использованием регулярных выражений до работы в сети. Библиотечные инструменты языка Python - это то место, где выполняется большая часть операций.

Программы на языке Python с легкостью могут «склеиваться» с компонентами, написанными на других языках программирования. Например, прикладной интерфейс С API в Python позволяет программам на языке С вызывать и быть вызываемыми из программ на языке Python. Из этого следует, что вы можете расширять возможности программ на языке Python и использовать программный код на языке Python в других языковых средах и системах.

Возможность смешивать Python с библиотеками, написанными на таких языках, как С или C++, например, превращает его в удобный язык для создания интерфейсов к этим библиотекам и в средство настройки программных продуктов. Как уже говорилось выше, все это делает Python прекрасным средством разработки прототипов - система может быть сначала реализована на языке Python, чтобы повысить скорость разработки, а позднее в зависимости от требований к производительности системы по частям перенесена на язык С.

Чтобы запустить программу на языке Python, достаточно просто ввести ее имя. Не требуется выполнять промежуточную компиляцию и связывание, как это делается в языках программирования, подобных С или C++.

Полезные источники:

Марк Лутц – Изучаем Питон

https://www.shashkovs.ru/\_prog/Lutc\_M.\_-\_Izuchaem\_Python\_(4-e\_izdanie)-\_2011.pdf

Курс лекций по Python от Google

https://www.youtube.com/watch?v=tKTZoB2Vjuk&list=PLC8825D0450647509