ИСТОРИЯ СОЗДАНИЯ ЯЗЫКА PYTHON

«Основы программирования». 10 класс

История создания

- Язык начал разрабатывать программист, Гвидо ван Россумом, в конце 1980-х. На тот момент он работал в центре математики и информатике в Нидерландах

- Гвидо ван Россум увлекался работой с «железками» ещё со школьных лет, и хотя он не находил поддержки и одобрения у своих сверстников, это не помешало ему самостоятельно разработать язык программирования

- Россум работал над Python в свободное время, в качестве основы он взял язык программирования ABC, в разработке которого когда-то участвовал

Этапы создания:

- Этапы истории языка программирования Python:
- В феврале 1991 исходный код языка был опубликован на alt.sources. Уже тогда язык придерживался объектно-ориентированного подхода, мог работать с классами, наследованием, функциями, обработкой исключений и всеми основными структурами данных

- В 2000 году вышла в релиз вторая версия Python. В неё добавили много важных инструментов, включая поддержку Юникода и сборщик мусора

- 3 декабря 2008 в релиз вышла третья версия Python, которая является основной до сих пор. Многие особенности языка были переделаны и стали несовместимы с предыдущими версиями. И хотя функциональность третьей версии ничем не уступает второй, развитие языка разделилось на две ветки. Кто-то продолжал использовать Python 2, чтобы поддерживать старые проекты, кто-то полностью перешёл на третью версию

Python — простой язык

- Синтаксис Питона всегда выделял его на фоне других языков программирования. Он не страдает избыточностью, схожесть синтаксиса с обычным английским позволяет понять код даже обычному пользователю, кроме того, программист пишет меньше строк кода, потому что нет необходимости использовать символы: «;», «{», «}». Вложенность обозначается отступами, что повышает читаемость кода и приучает новичков к правильному оформлению

- Простота отчасти обусловлена тем, что Питон написан на основе языка ABC, который использовался для обучения программированию и повседневной работы людей, не являющихся программистами

Особенности языка:

- **Динамическая типизация.** Программисту не нужно указывать тип переменных, язык присвоит его сам. Операнды разных типов, участвующие в одной операции, автоматически приводится к нужному по определённым правилам

- **Удобный возврат нескольких значений функцией.** Их можно перечислить через запятую и они автоматически преобразуются в список. Чтобы вернуть массив из функции, достаточно написать «return имя_массива«. Не нужно выделять память и передавать указатели в функцию

- **Автоматическое выделение памяти.** Программисту не нужно самостоятельно выделять память под что-либо. С одной стороны это уменьшает контроль программиста над программой, с другой, разработка значительно ускоряется

- **Сборщик мусора**. Если объект становится бесполезным (на него перестаёт что-либо ссылаться), он автоматически удаляется сборщиком мусора. Сборщик мусора позволяет оптимизировано использовать память и не удалять бесполезные объекты вручную
- **a, b = b, a**. Эта строка меняет местами значения переменных, теперь то, что было в a, находится в b и наоборот. Такое возможно, потому что Питон сначала рассматривает переменные справа от знака «=» и помещает их в список, то же он делает с элементами слева от «=», затем он связывает каждый элемент правого списка с левым. Таким способом можно обменивать значения не только двух переменных, но и трёх, пяти и так далее.
- **Привязка типа данных**. Тип данных привязан к значению, а не к переменной. То есть значение — это какой-то объект с атрибутами, которые определяют его тип и другие характеристики, а переменная — просто ссылка на этот объект. Такой подход позволил обойтись без явного определения типов и значительно упростил повторное присваивание значения переменной (особенно, если тип нового значения отличен от начального)

- **Цикл for.** Работать с массивами, списками и другими контейнерами в Питоне просто и удобно. Когда необходимо перебрать все его элементы, конструкция выглядит так: «for x in контейнер:» (перебор идёт от 0 до последнего элемента, его индекс можно обозначить как -1). Если нужно, чтобы прошло определённое количество циклов, пишут так: «for x in range(1,9):» (цикл будет выполняться со значениями x от 1 до 8)

- **Интерпретируемый язык.** Написанный код не нужно компилировать, достаточно запустить его и получить результат. Более того, можно работать в интерактивном режиме и получать результат буквально после каждой операции

- Python сочетает в себе и простоту и мощный инструментарий. Его можно использовать для создания прототипа практически любой программы

Приёмы, позволяющие нивелировать недостаточную скорость:

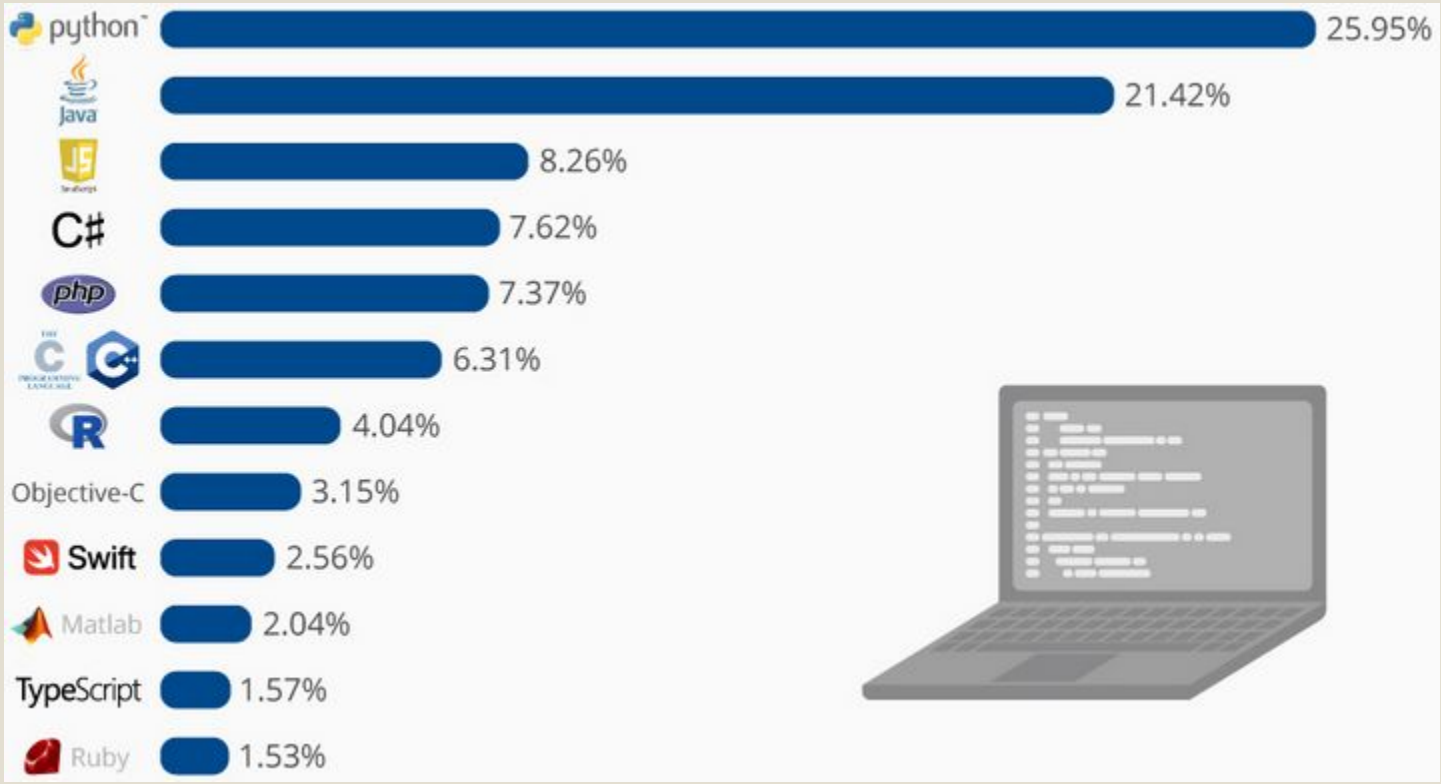
- **Встраивание кода на С**. С помощью такого приёма можно заметно повысить производительность, обычно на С пишут те участки кода, которые обрабатывают много запросов в единицу времени. Например, функцию, которая получает данные из одной базы данных, обрабатывает их и отсылает в другую, лучше написать на языке С, если объем проходящей информации достаточно большой.
- **Использование лучших алгоритмов и инструментов**. Одну и ту же задачу можно решить по-разному. Во-первых, программист должен выбрать наиболее эффективный алгоритм, обеспечивающий лучшую производительность, например, для поиска элемента в отсортированном массиве можно перебирать его от начала до конца, в лучшем случае (элемент в начале массива) поиск выполнится быстро, в худшем (элемент в конце массива) — медленно. Эффективнее использовать методом деления пополам (двоичный поиск), который найдёт нужный элемент за минимальное количество итерация в массиве любой длины. Во-вторых, для реализации задачи нужно подбирать правильные инструменты. Например, если последовательность элементов строго определена и не изменяется, лучше использовать кортеж, а не список. Он требует меньше места, обрабатывается быстрее и защищён от случайных изменений

- **Модули для тестирования**. Чтобы определить какие участки кода сильно снижают общую производительность, программист может использовать специальные модули для тестирования. Таким образом, можно понять, какой код нужно оптимизировать или заменить на код на языке C

Быстрее	Медленнее
a, b = c, d	a = c; b = d
a < b < c	a < b and b < c
not not a	bool(a)
a = 5	a = 2 + 3
код, встроенный в цикл	вызов функции с кодом в цикле

- **Готовые инструменты**. Для большинства задач уже разработаны эффективные решения. Лучше использовать готовый, отлаженный код какой-либо библиотеки, чем писать своё решение с нуля, которое 100% будет не таким эффективным

Популярность:



Что можно написать на Python?

Back-end сайта

- **Для разработки серверной части сайта используются фреймворки: Django и Flask.** Они превращают Python в серверный язык программирования, возможности которого не уступают другим популярным инструментам

- Программист легко может работать со связями URL адресов, обращениями к базам данных и созданием HTML файлов, которые пользователь видит в браузере

Blockchain

- Блокчейн — это последовательная цепочка блоков, где каждый блок содержит информацию и всегда связан с предыдущим. Технология может использоваться в любых сферах и особенно популярна в финансовой сфере и в сфере криптовалюты биткоин

- Блокчейн совмещает в себе защищенность и открытость информации, он позволяет получить доступ к данным из любой точки мира, в то же время его практически невозможно взломать, данные хранятся на каком-то главном компьютере, а взламывать каждый блок очень затратно и долго

- На Питоне без проблем можно написать полноценный блокчейн, если его грамотно спроектировать, то он не будет отставать по производительности от решений на других языках

Бот

- Это программа, автоматически выполняющая какие-либо действия в заданное время или в ответ на поступивший сигнал. Боты могут примитивно симулировать поведение человека, поэтому они часто используются для работы в технической поддержке (чат-боты), поиска информации в интернете (поисковые боты), имитации действий человека или другого существа в виртуальном мире (компьютерные игры)

- Python позволяет быстро создавать многофункциональных и относительно умных ботов. Важно понимать, что боты — это не простая программа в 500 строк кода. Заказ на создание бота для бизнеса может стоить несколько миллионов. Цена обусловлена тем, что спроектировать бота, которого будет сложно отличить от человека, очень сложно. Необходимо предусмотреть множество вариантов диалогов, проанализировать поведенческие факторы человека и внедрить их в программу. Проще говоря, из машины, понимающей только нули и единицы, нужно сделать примитивный «мозг»

База данных

- База данных — это информация, систематизированная по общим признакам и специальным правилам. В любом большом проекте используются базы данных, в них хранится информацию о пользователях, изменениях в программе и т. Д

- Систему управления базами данных можно написать на Python

Дополненная реальность

- Дополненная реальность дополняет физический мир с помощью виртуальных технологий. То есть виртуальные объекты проецируются на реальное окружение, и имитируют признаки и поведение обычных физических объектов
- Дополненную реальность можно наблюдать в фильмах, таких как Железный Человек. В реальном мире она используется, например, в боевых истребителях (система прицеливания)
- Работа дополненной реальности основана на взаимодействии с метками. Электронное устройство получает информацию и анализирует окружающее пространство, с помощью компьютерного зрения он «понимает», что человек видит перед собой. Затем устройство накладывает на реальный мир «виртуальный слой»
- Профессиональные приложения дополненной реальности стоят около полумиллиона рублей, проектировать и писать их не просто, в процесс разработки привлекаются различные специалисты, от 3D-дизайнеров до программистов
- Python является отличным инструментом для создания проектов дополненной реальности

BitTorrent клиент

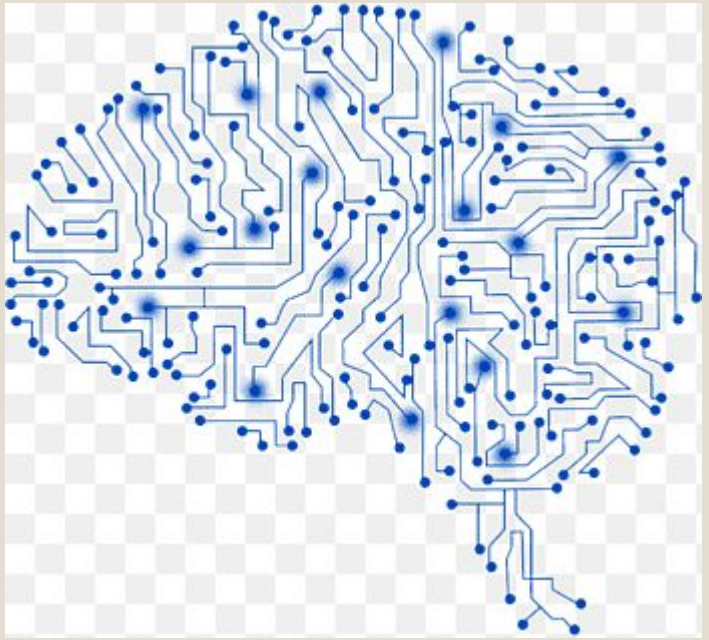
- BitTorrent — уникальная технология, позволяющая быстро обмениваться большими объёмами данных через интернет

- До 6 версии клиент BitTorrent был полностью написан на Python. И хотя позже он был полностью переписан на C++, это показывает, что Pyton можно использовать для реализации задач такого рода

Нейронная сеть

- Понятие «нейронная сеть» пришло в программирование из биологии. В биологии нейронная сеть — это последовательность нейронов, соединённых между собой. Программно созданные нейронные сети способны не только анализировать и запоминать информацию, но и воспроизводить её из памяти

- Они используются для решения сложных задач, где необходимы вычисления, которые совершаются человеческим мозгом.. Обычно нейронные сети используются для классификации чего-то по признакам, прогнозирования, распознавания, например, человека по фото или видео



Парсер

- Это ПО для сбора и обработки информации. Можно парсить такую информацию, как курс доллара, а можно следить и анализировать изменения акций различных компаний

- Парсер можно написать на многих языках, Python — не единственный хороший инструмент для этого, однако его возможностей вполне достаточно, чтобы написать приложение, собирающее информацию быстро и эффективно

Калькулятор

- Это задание выполнял, пожалуй, каждый студент факультета информатики. Калькулятор можно написать на любом языке программирования, и Python — не исключение

- Важно понимать, что от калькулятора требуется 100% точность расчёта. Поэтому все ошибки, связанные с округлением и двоичным представлением чисел могут быть критичными. Однако для Python написаны библиотеки, полностью решающие данную проблему

Игра

- На Питоне не создаются большие игры, он либо используется для разработки прототипа, либо для реализации какой-то части (например, серверной логики игры или системы моддинга)

- Для написания небольшого проекта можно воспользоваться библиотекой Pygame, которая даёт все необходимые инструменты для создания небольшой 2D игры

Текстовый редактор

- Он может использоваться для написания и редактирования не только текста, но и кода. Многие текстовые редакторы способны определять используемый язык программирования и подсвечивать его синтаксис. Некоторые из них и вовсе напоминают полноценную IDE

- Написать небольшой текстовый редактор не сложно, однако для создания крупного проекта понадобиться много знаний и сил. Несмотря на быстроту разработки на Python, создание текстового редактора с достаточным по современным меркам функционалам — это работа для целой команды программистов

Язык программирования

- Компьютер — это всегда многоуровневое устройство. Используя самый сложный и неудобный инструмент, программист создаёт более простой, а из него ещё более простой. Хотя это понижает производительность (если бы всё было написано на ассемблере, программы работали бы в десятки или даже сотни раз быстрее), но также и значительно уменьшает время разработки, её удобство и сложность

- Python достаточно высокоуровневый язык, поэтому писать на его основе ещё один язык программирования нецелесообразно, хотя и можно. Полезнее будет разработать интерпретатор для самого Python или другого языка программирования. Также можно создать компилятор (программа, конвертирующая код языка программирования в машинный)

- Подобные проекты не подойдут для коммерческих целей, но создание своего компилятора, интерпретатора или языка даст много бесценного опыта