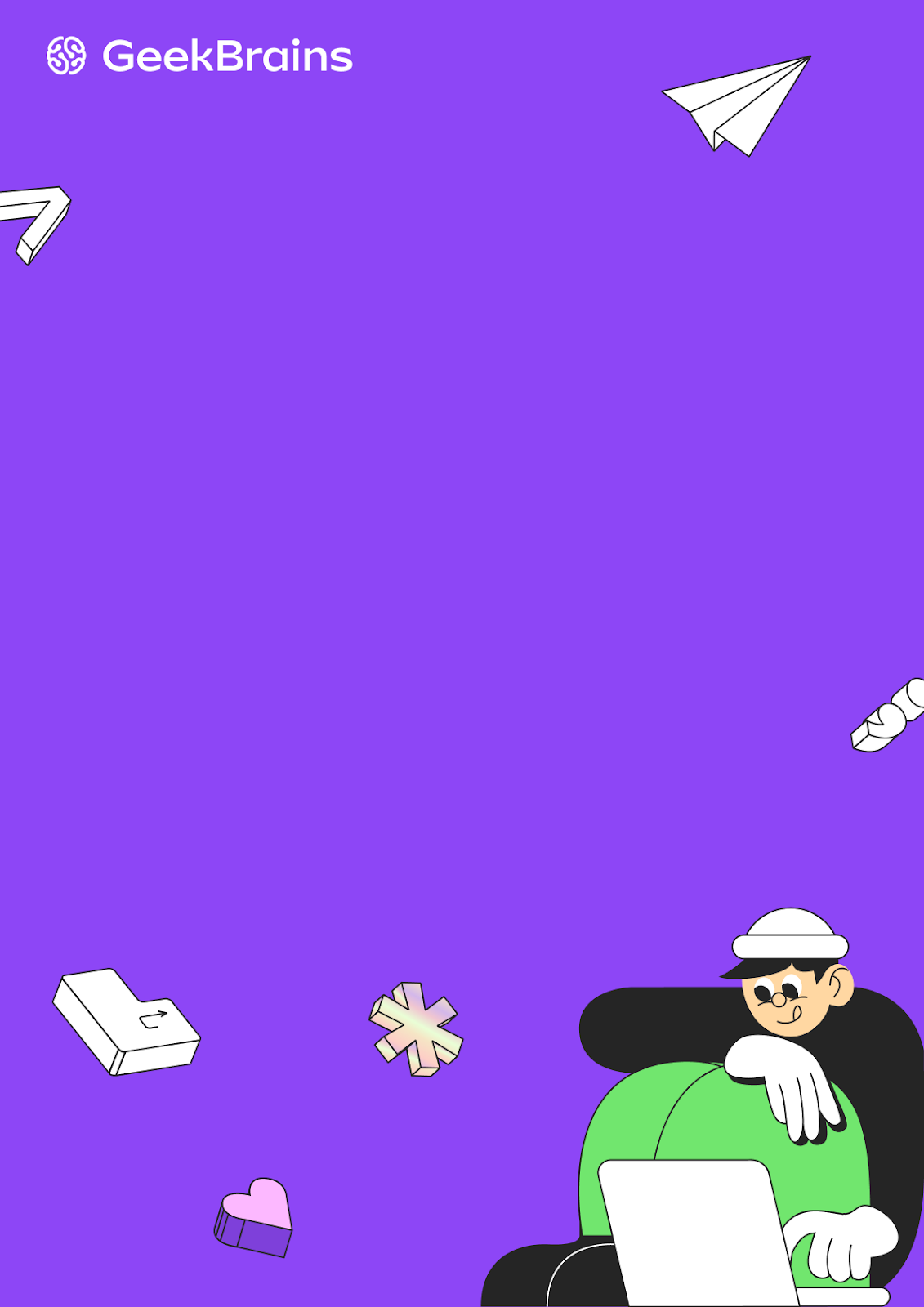
****

### Специальность: «Python - разработчик» — 3246

### Тема дипломной работы:

**Разработка Telegram-бота для парсинга интернет-магазинов.**

### ФИО студента – Таганов Валерий Иванович

Ссылка на проект: [ParcerBot (telegram.org)](https://web.telegram.org/a/#6358820839)  
Ссылка на GitHub [Valerii21 (github.com)](https://github.com/Valerii21)

Web версия дипломной работы находиться по адресу:  
[Valerii21/Diplom-GeekBrains (github.com)](https://github.com/Valerii21/Diplom-GeekBrains/tree/main)

#### Место и год написания - Москва, 2023

1. **Введение**  
 Описание проблемы и актуальности темы  
 Цели и задачи проекта

2. **Основная часть**

**Теоретическая часть**  
 Описание интернет-магазинов

Выбор языка программирования

Анализ методов парсинга

Обзор библиотек для работы с Telegram

Обзор методов и технологий, используемых в работе

Python

pyCharm

Bot Api

Aiogram

Beautiful soup

**Практическая часть**

Разработка алгоритма парсинга

Создание Telegram-бота

Тестирование Telegram-бота

Оценка эффективности работы бота

3. **Заключение**  
 Основные результаты проекта

Выводы и рекомендации по использованию телеграмм бота в практике  
 Перспективы развития  
 Список использованных источников  
**4.Приложения**  
 Код скрипта на Python  
 Скриншот Telegram-бота

**Введение   
Описание проблемы и актуальности темы**

В настоящее время все больше компаний и частных лиц используют интернет для поиска информации, заказа товаров и услуг, а также для общения. Однако, чтобы эффективно использовать интернет, необходимо уметь быстро и точно извлекать информацию с сайтов и парсить ее. Ручной парсинг сайтов является трудоемким и требует много времени, поэтому многие компании и частные лица предпочитают использовать специальные программы или скрипты для автоматизации этого процесса.

Одним из таких инструментов являются телеграмм боты - программы, созданные на основе мессенджера Telegram, которые способны выполнять различные задачи, в том числе парсить сайты. Телеграм-бот позволяет быстро и легко получать информацию с любого сайта независимо от его сложности и структуры, что значительно ускоряет процесс парсинга.

Однако, несмотря на все преимущества, использование телеграмм бота для парсинга сайтов не является достаточно распространенным явлением. Многие компании и частные лица не знают, как создать телеграмм бота и настроить его для работы с парсером. Кроме того, существует проблема обработки полученной информации, что может привести к ошибкам и неправильным результатам.

Некоторые сайты имеют сложную структуру, которая не может быть полностью обработана с помощью телеграмм ботов без дополнительных инструментов и знаний. Кроме того, некоторые сайты могут использовать защиту от парсинга, что также может затруднить работу телеграмм ботов.

Все это делает разработку телеграмм бота для парсинга сайтов актуальной и важной задачей. Создание такого бота поможет компаниям и частным лицам быстро и точно получать нужную информацию, сокращая время на ее поиск и обработку. Кроме того, использование телеграмм ботов для парсинга сайтов может быть полезным для анализа рынка, определения тенденций и принятия решений на основе актуальной информации.

**Цели и задачи проекта**

**Цели проекта:**

Разработать телеграмм бота для парсинга сайтов;

Научить телеграмм бота парсить веб-страницы;

Провести тестирование и анализ работы телеграмм бота.

**Задачи проекта:**

Изучить методы парсинга веб-страниц;

Выбрать язык программирования для создания телеграмм бота;

Настроить бота и подключить парсер;

Написать код для реализации алгоритмов парсинга и обработки данных;

Протестировать телеграмм бота на различных сайтах;

Оценить качество и точность работы бота;

Реализовать хранение данных и доступ к ним;

**Основная часть**

**Описание интернет-магазинов**ТОП Маркетплейсов России: список лучших, рейтинг 2023 года

Рейтинг российских интернет-магазинов в 2023 году с точки зрения покупателей. Анализировались все подводные камни, минусы и плюсы условий для покупателей, актуальные для июля 2023 г. Рейтинг основан на следующих критериях: дополнительные затраты покупателя, удобство покупателей, логистические, обороты, потери товара, аналитика.

Ozon

Из года в год маркетплейс «Озон» лидирует в рейтинге наиболее популярных российских торговых площадок, входит в топ-10. В конце 2022 г. число активных покупателей превысило 150 тыс. человек.

По обороту интернет-портал занимает 2-е место по сравнению с Wildberries, однако по конкуренции в настоящее время поставщиков на Озоне все равно меньше и потому конкуренция среди категорий выше. Сайт: ozon.ru

Плюсы:

1. Бесплатная регистрация на торговой площадке.

2. Прозрачная аналитика.

3. Есть бесплатная доставка товара.

4. Не очень часто происходят потери и порча товара.

5. Отзывчивая служба поддержки клиентов.

6. Гибкая система скидок.

Минусы:  
  
1. Долгая доставка товаров до покупателя.  
  
Wildberries

Wildberries является крупнейшим маркетплейсом в России, занимающим 13% рынка в 2022 году. По итогам 2022 года на площадке зарегистрированы покупатели из России, Беларуси, Казахстана, Словении. Количество покупателей - более 40 миллионов человек.

По обороту Wildberries занимает первое место среди маркетплейсов, однако по отзывам с натяжкой можно ставить только на второе. Сайт: wildberries.ru

Плюсы:

1. Удобный и простой личный кабинет.

2. Быстрая доставка до покупателя.

Минусы:

1. Частая порча товара.

2. Личный кабинет часто не работает.

3. Служба поддержки систематически игнорирует обращения.

3. Яндекс Маркет

Яндекс Маркет – это маркетинговый портал. Это как рыночная площадка или крупный супермаркет, в котором есть всё: от холодильника до косметики.

Продавцы на Маркете размещают свою продукцию и выставляют цены, Маркет находит покупателей.

Входит в топ-3 среди магазинов, менее популярен среди клиентов. Сейчас активно вкладываются средства в рекламные инструменты для того, чтобы достичь более высокого уровня. Сайт: market.yandex.ru

Плюсы:

1. Большая программа лояльности для покупателей.  
2. Низкая комиссия для дешевых товаров, для продавцов.  
3. Присутствует бесплатная регистрация на площадке как для покупателей, так и для продавцов.

Минусы:

1. Служба поддержки долго отвечает на обращения.

**Выбор языка программирования.**

Создание качественного, функционального бота в Telegram является трудоемким и непростым процессом. Для создания бота необходимо много знаний, навыков и выстраивание грамотной стратегии создания.

Многие специалисты считают не принципиальными различия между выбором языка создания бота в Telegram. Очень важно произвести коммуникацию с API-сервера.

По статистике, в большинстве случаев боты пишут на Python. Это достаточно легкий язык и у него много реализованных библиотеках. Кроме того, бот может быть написана с помощью Java, Ruby, JavaScript, Node.js, go и другие.

Лучше выбирать язык бота, подходящий для определенной задачи.

Самым часто используемым и современным языком программирования является Python. Особенно он пригодится тем, кто первый раз будет программировать. У языка простой синтаксис с большой стандартной библиотекой функций.

Для создания бота нужен пакет Python-Telegrambot - API-оболочка от Telegram.   
  
Результаты тестирования разработчиков ботов в Telegram.

На сегодняшний день в топе три языка - Python (41%), PHP (15%), NodeJS (11%).

Это говорит о том, что, изучая Python, будет намного больше шансов найти поддержку других разработчиков.

Низкая граница входа.

Python имеет очень простой и понятный синтаксис. В случае выбора Python потребуется минимум времени и усилий, для изучения этого языка программирования.

Большое число готовых решений.

С учетом низкого порога входа и популярности языка Python, сегодня существует тысячи готовых библиотек и ежедневно появляются новые.

**Анализ методов парсинга**

В переводе «парсинг» означает грамматический разбор слова или текста. Изначальный смысл произошел от латинского «парс ораторий» - части слова. Поэтому парсинг – это метод, при котором над строкой или текстом проводят анализ и разделяют на компоненты. Затем полученная информация преобразуется в подходящий формат, чтобы обработка и использование было более удобной в последующей работе.

Оказывается, один формат информации превращается в другой формат, более понятный человеку. Обычно получают данные в чистом HTML-коде, парсер принимает и конвертирует их в формат, который легко анализировать и понимать.

Получается, что один формат данных преобразуется в другой формат данных, более удобно читаемый для человека. Обычно данные получают в HTML, а парсер принимает его(обычно всю страницу целиком) и преобразует в формат, который можно легко проанализировать и понять.

Полученные данные хранятся в отдельных файлах – либо локальных компьютерах, либо облаках или хостингах, либо прямо в базе данных. Это автоматический процесс, выбираемый и прописываемый программистом.

Программное обеспечение помогает анализировать собранные данные. Работающий парсер посылает на сайт запрос типа GET, который взамен должен отдать ему данные. В результате данного запроса создается HTML документ, который проанализирует программа. После этого парсер ищет в нем необходимые данные и преобразует их в соответствующий формат.

Существует 2 различных подхода выполнения веб-парсинга:

• доступ к сайту через протоколы HTTP, HTTPS или веб-браузер;

• доступ с использованием бота (обычно используют Телеграм боты).

Парсинг — это разрешенный метод сбора и анализа информации. Иногда, на сайтах государственных структур предоставляются данные для публичного использования, доступные с помощью API. Поскольку такая информация требует обработки огромного количества данных, парсеры приходят на помощь.

Для проведения анализа заданных текстов ПО, как правило, используется отдельный анализатор лексики. Это токенайзер или лексер. В информатике лексический анализ («токенизация», от англ. tokenizing) — процесс аналитического разбора входной последовательности символов на распознанные группы — лексемы — с целью получения на выходе идентифицированных последовательностей, называемых «токенами» Лексер разбивает все данные на токены, например отдельные символы или слова. В итоге, полученные ключевые данные являются входными символами для обработки кода парсером.

Потом программа делает обработку грамматики входных данных, анализирует их, после чего создает синтаксическое дерево. Основываясь на этих данных, идет последующая работа парсера с информацией. Генерация кода или выборка по заранее определенным нами критериям (токенам или ключевым словам).

Есть 2 типовых вида парсинга: сверху-вниз и снизу-вверх. Они различаются направлением движения по документу, в котором создаются узлы синтаксического дерева.

* Сверху-вниз: при нисходящем методе парсер выполняет поиск сверху — с начального символа в коде сайта ищет подходящие ему синтаксические связи. В итоге, дерево синтаксического анализа растет сверху вниз, в направлении более детальной разбивки.
* Снизу-вверх: восходящий парсер начинает снизу, с самого нижнего символа строки, а затем строит все более крупные синтаксические связи. Это делается до тех пор, пока не будет достигнут изначальный символ кода.

Метод обработки кода не самое важное. Хороший работающий парсер, восходящий или нисходящий, будет отличать, какую информацию нужно найти в коде HTML-страницы. И, согласно заранее написанному коду и правилам синтаксиса, выбирает интересующие нас данные, преобразует их, к примеру, в формате JSON, CSV или в Excel-таблицу.

Отмечу, что сам парсер не привязан к какому-либо формату. Парсер только инструмент, преобразующий один формат в другой. Как он преобразует данные и во что, будет зависеть только от стоящей перед нами задачи.

Парсинг применяется для перевода текста в новый формат в следующих ситуациях:

* Чтение кода программного обеспечения, например Java, SQL, другие языки. Это делают парсеры баз. Парсер дает программному компилятору структуру данных, которая позволяет сгенерировать машинный код.
* Читать HTML-код с помощью исходного кода страницы парсера.   
  HTML код для компьютера - это только строки символов, которые нужно проанализировать парсером браузера. Парсер дает описание страницы как структуры информации, которая потом компонуется в соответствии с необходимым порядком и будет выведена на экран ПК.
* Анализ разметки XML. Специальный парсер XML отвечает за анализ данного вида документов, подготавливает содержимое в ней информации для их дальнейшего применения.
* Чтение URL адресов и протоколов интернета HTTP и HTTPS. Парсеры разбивают сложные URL-адреса и протоколы на их иерархическую структуру.
* Поисковая система.   
  При помощи поисковых парсеров боты выбирают тот текст, который они ищут на странице сайта. После первого выбора полученные данные обрабатываются и анализируются, согласно правилам, которые мы задали парсеру, а результаты анализа становятся доступны для просмотра.

Парсинг используется в цифровом маркетинге для сбора и анализа определенной информации из контента нужного сайта.

Парсинг страниц является основным средством сбора информации из контента веб-сайтов, который используется в разных сферах – продажах, маркетинге, финансах, электронной торговле, сборе информации о конкурентах и т.д. Активно применяется в следующих сферах.

Розничная торговля

В розничной торговле существует много возможностей использования парсинга. К примеру, мониторинг цен или анализ рынка, где парсинг используют для обработки данных и извлечения из них наиболее ценной информации.

Анализ рынка акций

Раньше анализ рынка фондовых бумаг ограничивался изучением финансовых отчетов компаний, а соответственно инвестированием в самые подходящие акции. Сегодня все новости, изменения в политике или обществе важны, чтобы определить текущие тренды. Парсинг помогает получить альтернативные данные. Он позволяет получать всю информацию о рынке и видеть общую картину.

Машинное обучение

Обучение проходит каждый парсер. Это дает возможность искусственному интеллекту обнаружить закономерность. Но для установления нужных связей нужно передать много данных в разум компьютера и помочь связать одну информацию, с другой. Часто парсеры используются в AI-технологиях для обеспечения регулярного потока учебной информации.

Парсер почты

Парсинг электронных сообщений позволяет проанализировать входящие и исходящие письма. После чего их содержимое можно интегрировать в различные приложения или сайты c помощью программного интерфейса API, отсеять в спам или собрать для дальнейшего анализа.

Большая часть компаний использует API-решения для трех основных типов приложений.

1. Поддержка заказчиков. Парсинг позволяет фиксировать взаимодействие с клиентами через электронную почту, отправлять нужные сообщения пользователям, собирать и анализировать входящие сообщения, чтобы хранить и отображать в поддержке приложения.

2. Приложения по управлению отношениями с клиентами в CRM. В CRM-приложениях часто записываются все переписки с клиентом для сохранения ценных данных из истории сообщений. При отправке ответов клиентов по службе парсинга приложение CRM может получить предварительно анализированные данные.

3. Социальное приложение. Сайты социальных сетей часто дают возможность общаться через электронную почту для того, чтобы пользователи могли отслеживать беседу. Парсинг входящей почты является простым и быстрым способом настроить приложения почты так, что они будут расширяться в зависимости от масштаба переписки в социальных сетях.

Принцип работы парсеров для сайтов одинаков, как правило он состоит из 3 этапов.

1. Запрос-ответ

Первым шагом является поиск содержимого целевого сайта в соответствии с определенным URL-адресом. Взамен пользователь получает запрошенную информацию в HTML-формате.

2. Разбор и извлечение

Синтаксические анализы обычно используются для любого компьютерного языка. Это процесс распознавания кода как текста, создания в памяти структуры, с которой компьютер может разобраться и уже работать с ним.

Парсер берёт HTML код и берет оттуда соответствующие данные - например, заголовок сайта, абзац, подзаголовок, ссылку, выделение жирного, нужные темы и т.д., а также проводит парсинг текстов.

3. Загрузка данных

Полученная информация загружается и сохраняется. Задается формат файла, чтобы открыть его в другом нужном приложении. Для Таблиц Google это CSV, парсинг баз данных JSON, и т.д.

**Обзор библиотек для работы с Telegram**

Одна из самых лучших Python библиотек для работать с Telegram - aiogarm. Она предназначена для того, чтобы создавать боты в Telegram и обладает высоким уровнем производительности и поддержки. Однако у aiogram могут быть проблемы с документами, но в Telegram есть чат, где можно всегда получить поддержку от разработчиков библиотек.

Если требуется что-нибудь простое с множеством хэндлеров и не хватает времени для изучения, можно воспользоваться библиотекой Python-Telegrambot. У нее подробная инструкция, шаблоны кода, поддерживает синхронизацию при помощи декоратора Run Async.

В настоящее время библиотек для работы с Telegram очень много. Они могут быть разбиты по виду выполняемых задач:

1. Библиотеки для того, чтобы создать и отправлять сообщения в Telegram.

2. Библиотеки, позволяющие получить информацию по каналам, группам и пользователям.

3. Библиотеки для получения уведомлений о новых сообщениях и других событиях в Telegram.

4. Библиотеки для работы с Telegram API и интеграции Telegram с другими приложениями.

5. Библиотеки с открытым исходным кодом и библиотеки с коммерческими лицензиями.

6. Библиотеки для работы с различными языками программирования, например Python, Java, C, Ruby и т.д.

7. Библиотеки разных компаний, которые можно отличить по функционалу, удобству в использовании и поддержки.

8. Библиотеки различной сложности, функциональности, предназначенные для различных задач и целей.

9. Библиотеки поддержки работы с Telegram на различных платформах и устройствах.

В целом выбор библиотеки, которая будет работать с Telegram, зависит от определенных требований и задач.

**Обзор методов и технологий, используемых в работе**

**Python**

Python является языком программирования, массово использующимся в мировом сообществе программистов, в разработке ПО, в науке, обработке данных, в machine learning. Программисты используют Python, так как он эффективен и прост в изучении, понимании, работает на различных платформах. Скачать программу на Python всегда можно бесплатно, она совместима со всеми распространёнными операционными системами и увеличивает скорость разработки необходимых для нас программных продуктов.

В Python есть интересующие нас преимущества:

Программист может с легкостью прочитать и понять программу написанную на Python, так как язык имеет очень простой синтаксис.

Python дает возможность программистам работать более продуктивно, так как появляется возможность писать программы на Python используя меньшее количество строк кода, чем на других языках.

В Python уже есть большая стандартная библиотека, содержащая многократно использованные до нас коды, практически для любой задачи. Мы с легкостью можем найти уже готовый исходный код. В результате не придётся писать код с нуля.

Можно с легкостью сочетать Python и другие языки программирования, такие как: Java, C, C++ и другие.

Сообщество Python включает в себя миллионы активных разработчиков по всему миру. В случае возникновения ошибки, активные разработчики могут помочь ее решить.

Также в Интернете есть много полезных источников для ознакомления с Python. Например, можно легко найти фильмы, учебники, документы и инструкции для разработчиков.

Программа на Python может быть перенесена на любые операционные системы: Windows, MacOS, Linux.

Для разработки приложений на Python существует несколько широко известных стандартов использования, в том числе:

Веб-разработка на стороне сервера

Веб-разработка на стороне сервера учитывает сложные функции серверов, которые позволяют сайтам отображать необходимую информацию пользователям. Например, почти все сайты взаимодействуют с базой данных, иногда с другими сайтами, а также защищают данные при отправке.

Python может быть использован для написания серверных приложений, так как в нем присутствует много библиотек с предварительно написанными кодами для выполнения на стороне сервера. Также можно использовать широкий ассортимент платформ Python, предоставляющих все необходимые средства для быстрой и простой разработки интернет-приложений.

Автоматизация с помощью скриптов Python

Язык скриптов является языком программирования, автоматизирующем задачи, которыми обычно занимаются люди. Программисты используют Python-скрипты, чтобы автоматизировать многие повседневные процессы, такие как:

* Необходимость одновременного переименования большого количество файлов
* Смена типа файлов в другой тип
* Удаление повторяющихся слов в текстовом файле
* Выполнение математической операции
* Работа с электронной почтой
* Скачивание и загрузка контента.
* Работа с аналитикой

Наука о данных и машинное обучение (machine learning)  
 При научной работе с данными извлекается наиболее важная информация, а машинное обучение (ML) позволяет компьютерам автоматически анализировать и учиться на основе этой информации. Пример использования: прогнозы в различных областях - погода, акции. Специалисты по работе с данными используют Python для решения всевозможных задач:

Редактирование данных и исправление ошибок в них  
Изучение и выборка данных по различным параметрам  
Разметка данных   
Поиск статистической информации и работа с ней  
Визуализация при помощи диаграмм и графиков

Специалисты по работе с данными обычно используют Python ML библиотеки для создания моделей и классификаторов для точного разбиения данных. На основе Python классификаторы используются в различных сферах и применяются для решения многих задач, таких как классификация изображений, текстов и различного трафика, распознавание лиц. Специалисты по работе с данными также используют Python для глубокого обучения — передовой техники машинного обучения.

Разработка программного обеспечения

Программисты часто используют Python для всевозможных задач и программных решений, среди которых:

* Поиск ошибок в коде ПО
* Сборка ПО
* Управление проектами
* Проектирование шаблонов ПО
* Разработка настольных приложений с использованием GPI (графический пользовательский интерфейс)
* Создание игр

Автоматизация тестирования ПО

Тестирование ПО является процессом проверки ожидаемых результатов работы программных продуктов и отсутствие ошибок в них.

Программисты используют различные среды модульного тестирования Python (Unit Test, Robot Test и Python Unit Test) для того, чтобы тестировать получившиеся программные продукты.

Тестировщики иногда используют Python, для написания тестовых скриптов для последующей работы с различными сценариями. Python используется для проверки пользовательских интерфейсов интернет-приложений, программных компонентов и новых функций. Как правило все скрипты уже написаны и их можно найти в открытом доступе, но тестировщики все равно должны разбираться в коде.

Тестировщик может запустить тестовые скрипты несколькими способами. Такие инструменты называются инструментами непрерывной интеграции или непрерывного развертывания (CI/CD). Тестировщики и разработчики ПО используют CI/CD для автоматизации процессов тестирования. Инструмент CI/CD автоматически запускает тестовые скрипты Python и сообщает о результатах тестирования всякий раз, когда разработчики вносят какие-то изменения в проект.

Развитие Python

Python разработан Гвидо Ван Россумом (Guido Van Rossum), программистом из Нидерландов. Он начал работу над языком еще в 1989 году в центре Centrum Wiskunde & Informatica (CWI). Изначально язык был полностью любительским проектом: Ван Россум просто хотел чем-то занять себя на рождественских каникулах. Название языка было взято из телешоу BBC «Летающий цирк Монти Пайтона», большим поклонником которого являлся программист. 

История версий Python

Gvido Van Rossum выпустил первую Python-версию в 1991 г. Это уже включало в себя множество полезных функций. Например, существовали различные виды данных, а также функции, которые могут обрабатывать ошибки.

В версия Python 1.0, выпущенной в 1994 г., была реализована новая функция для обработки списков данных: сопоставление, фильтрация и сокращение.

Python 2.0 была выпущена 16 Октября 2000 года, в которой были представлены новые полезные функции для программистов, например поддержка Unicode символов и упрощенный способ циклического просмотра списка.

Python 3.0 вышел 3 декабря 2008. В этой версии была включена функция печати, а также дополнительная поддержка деления чисел и обработки ошибок.

Особенности Python

Язык Python единственный в своем роде, благодаря следующим особенностям:

Интерпретируемый язык

Python - интерпретируемый язык, т.е. код выполняется построчно. Если в коде есть ошибки, то программа перестает работать. При таком подходе всегда можно легко и быстро найти ошибку.

Простой в использовании язык  
Язык Python очень прост, так же прост и его синтаксис. К примеру: в Python не используется фигурная скобка. Вместо этого применяют отступы.

Язык с динамической типизацией

Программисты не должны объявлять тип переменных в процессе программирования, так как Python сам определяет тип в процессе выполнения кода. Эта функция значительно упрощает работу.

Язык высокого уровня

Python на много ближе к обычному языку, чем множество других языков программирования. Таким образом, программисты не должны волноваться за его основные функции, например, за архитектуру и управление памятью.   
Объектно-ориентированный язык

Python считает все объекты объектами, но поддерживает и другие виды программирования, такие как структурное и функциональное программирование.

Библиотеки Python

Библиотеки является наборами часто используемых программных кодов, которые разработчики могут включить в свои программы, для того чтобы не писать код с нуля. В Python по умолчанию есть большая стандартная библиотека с уже многократно используемыми возможностями для разработки.

Наиболее популярные библиотеки в Python

Matplotlib

MatPlotlib используют для отображения данных в высококачественных двух-и трехмерных графиках. Данную библиотеку обычно используют при решении научно-исследовательских задач. При помощи Маtplotlib на основе данных могут быть построены различные диаграммы и графики. Результат работы можно переносить на различные платформы.

Pandas

Pandas используют для управления данными временных рядов и структурированными данными, такими как таблицы и массивы. К примеру, при помощи Pandas можно читать, записывать, объединять, фильтровать и группировать данные. Также данная библиотека очень часто используется в науке о данных, анализе данных и задачах machine learning.

NumPy

NumPy используется для того, чтобы создавать массивы и управлять ими, управлять логикой и выполнять математические операции над ними. NumPy поддерживает интеграцию со многими языками программирования.

Requests

Библиотека Reqests содержит в себе полезные функции и возможности для веб разработки.

OpenCV-Python

OpenCV Python используется для обработки изображений при работе с машинным зрением. Библиотека обладает различными функциями работы с изображениями. Используя данную библиотеку можно одновременно читать и записывать изображения, осуществлять преобразование двухмерной среды в трехмерную.

Keras

Keras это библиотека для создания и работы с нейронными сетями.

Платформы Python

Python-платформы - это пакеты и различные по функционалу модули. Модуль является набором связанных программных кодов, а пакет является набором модулей. Программисты могут использовать Python-платформы для быстрого создания и упрощения разработки приложений, так как не нужно держать в голов за низкоуровневые детали. К примеру: скорость обмена данными.  
  
 Python обладает двумя типами платформ:

Платформа полного стека, которая включает в себя почти все необходимое для создания и развертывания крупных приложений.

Микро платформа является базовой платформой, которая предоставляет минимальные функциональные возможности создания простых Python-приложений.

Популярные платформы Python

Чтобы сделать свою разработку более эффективной и функциональной, можно использовать несколько платформ Python. Например:

Django

Джанго является одной из самых популярных платформ, обладающих полным стеком Python, используемых для создания крупных интернет-приложений. В ней представлены некоторые полезные функции, среди которых веб-сервер, для разработки и тестирования, движок шаблонов для разработки Frontend и разные механизмы защиты.

Flask

Flask представляет собой микро платформу, которая позволяет создавать небольшие интернет-приложения. Ее особенностями являются сильная поддержка сообществом программистов, качественная документация, движок шаблонов, модульное тестирование, встроенный веб-сервер. Также на платформе имеются расширения, которые поддерживают валидацию, уровни отображения баз данных, а также веб-безопасность.

TurboGears

TurboGears представляет собой платформу, которая предназначена для создания более быстрых и простых интернет приложений. Ниже приводятся основные ее возможности.

Определенные структуры базовых данных

Инструменты по созданию и управлению проектами

Движок шаблонов, чтобы создать базы данных

Движок шаблонов для Frontend-разработки

Механизмы обеспечения безопасности веб-сайтов

Apache MXNet

Apache MXNet – это быстрая платформа глубокого обучения, которая создает исследовательские прототипы и приложения глубокого обучения. Платформа поддерживает несколько языков программирования, в том числе Java, C++, Perl. Платформа обладает богатым набором инструментов, библиотек для разработки. Например, в ней представлена книга о интерактивном машинном обучении ML, наборы инструментов для машинного зрения и модели глубокого обучения к обработке естественного языка NLP, включая текст и речь.

PyTorch

PyTorch – это платформа по машинному обучению, созданная по библиотеке Torch, другой библиотеке машинного обучения с открытыми исходным кодом. Разработчики используют ее в NLP, робототехнике и машинном зрении, чтобы искать важную информацию в изображениях, видео. Также используется платформа для запуска приложений на процессорах и графических процессорах.  
  
 IDE Python

IDE-среда для разработки является программным обеспечением, позволяющим разработчикам писать, редактировать, тестировать и отлаживать код.

Популярные Python IDE

PyCharm

PyCharm является результатом работы компании JetBrans, чешского разработчика программного обеспечения. Программа имеет бесплатную версию для маленьких приложений и платную профессиональную версию, которая подходит для разработки больших приложений Python с следующими функциями:

Автоматическое завершение и проверка кода

Обработка ошибок и быстрое их устранение

Чистка кода без изменения функционала

Поддержка платформ интернет-приложений, таких как Django и Flask   
  
Работает с другими языками программирования.

Научные инструменты и библиотеки, такие, как Matplotlib, NumPy.

Возможность запуска, отладки, тестирования, развертывания приложений в удаленной виртуальной машине

Отладчик по поиску ошибок в коде, профилировщик по выявлениям проблем с производительностью и средством запуска модульного тестирования.

Работа с базами данных

IDLE

Интегрированная среда разработки и обучения IDLE — это интегрированная среда разработки Python, которая установлена по умолчанию. Среда разработана только на Python с использованием определенного набора инструментов Tkinter GUI и имеет следующие особенности:

Совместима с многими операционными системами, такими как Windows, UNIX и MacOS

Командное окно для запуска команд и отображения вывода

Многооконный редактор текста с подсветкой синтаксиса кода и авто завершением.

Встроенное устройство для отладки

Spyder

Spyder представляет собой IDE с открытым исходным кодом, который используют множество экспертов и аналитиков. Spyder применяется для всесторонней разработки с использованием функций расширенного анализа данных, отладки и визуализации. Среда обладает следующими особенностями:

* Редактор кода, поддерживающий множество языков
* Интерактивная консоль IPython
* Базовый отладчик
* Научно-исследовательские библиотеки, например, Matplotlib, SciPy и NumPy
* Возможность исследовать переменные в коде.
* Возможность просмотра документации в режиме реальном времени

Atom

Atom является бесплатным редактором, разработанным GitHub, поддерживающий кодирование на различных языках, включая Python. Атом позволяет работать напрямую с GitHub, сайтом, где можно централизованно хранить собственный код. У Atom есть такие особенности, как:

* Совместимость с множеством операционных систем
* Простая установка или создание новых пакетов
* Быстрое автозавершение работы кода
* Возможность найти файлы и проекты
* Простой настраиваемый интерфейс.

Python SDK

Пакет программного обеспечения SDK является набором программных инструментов, позволяющий создавать программное обеспечение на определенной языковой основе. Большая часть SDK разработана для различных платформ для аппаратных операционных систем и платформ. Python SDK содержит много инструментов, например библиотеки, шаблоны кода, а также руководства разработчикам, помогающие в написании приложения.

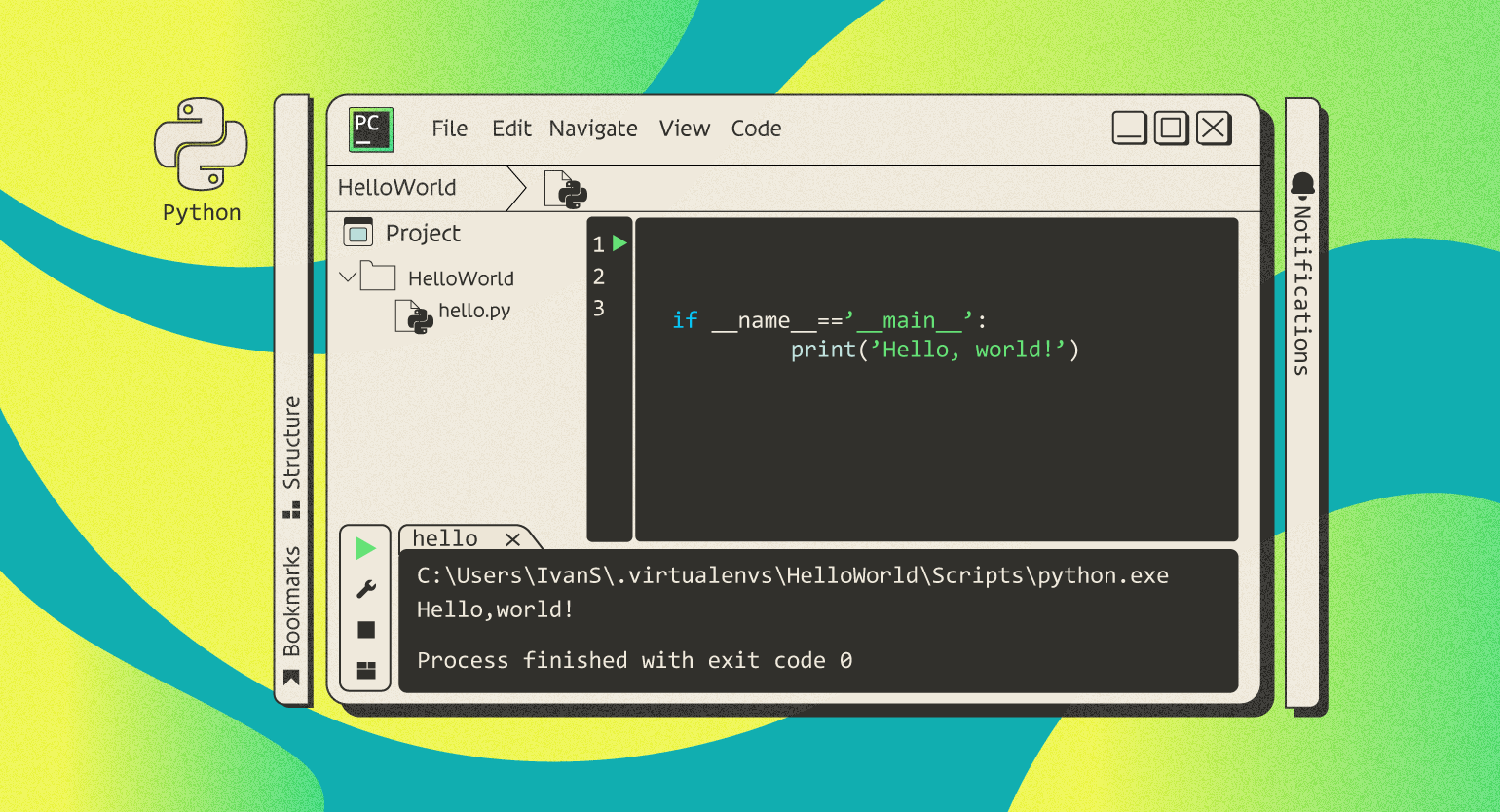
AWS PyCharm

Набор AWS-инструментов для PyCharm - это модуль для Pycharm IDE, который позволяет создавать, отлаживать и разворачивать приложения Python на AWS. AWS Toolkit for PyCharm. Программный комплекс AWS для PyCharm значительно упрощает начало Python-разработки. У него есть множество полезных возможностей для разработки, среди которых есть руководство по началу работ с набором инструментов, так же пошаговой отладке и развертыванию IDE.

Boto3 в Python

Boto3 – это SDK AWS for Python. Для создания и управления сервисами AWS, например, Amazon EC2, Amazon S3, Amazon Dynamo DB, можно использовать его для разработки и управления сервисов. Boto3 обладает 2 типами API: низкоуровневым и API-интерфейсы ресурсов для программистов.

**PyCharm**



Можно даже написать код в блокноте, и он будет работать. Однако делать это куда удобнее в специальной разработанной среде, которая будет помогать программировать быстрее, чище и проще. Один из редакторов — PyCharm, созданный для тех, кто пишет код на Python.

PyCharm является интегрированной средой разработки на Python, PyCharm разработала компания JetBrains. Среди программистов эта IDE является второй по популярности после Vs Code: в качестве основной ее используют 31% разработчиков.

В PyCharm имеются почти все инструменты для написания, отладки и тестирования кода. Например, программу можно сразу исправить в нескольких местах, а встроенное форматирование приведет ее в соответствие стандарту написания кода PEP8.

В IDE также можно использовать другие программные языки и синхронизировать код с системой управления версиями и разворачивать свой программный продукт.

Основные функции PyCharm могут быть расширены плагинами, позволяющими, к примеру, настроить интерфейс и подключить дополнительные инструменты.

Системные требования для PyCharm

Требования для операционной системы:

* Windows 8, 10 или 11 версии (64-битные);
* macOS 10.14 или более поздняя версия, в более ранних версиях не поддерживается;
* По настройке Linux лучше действовать методом проб и ошибок, так как информации в интернете много, но часто она противоречивая. Так же многое зависит от настроек вашего компьютера.
* Процессор не требует каких-то особых требований, однако рекомендуется использовать многоядерный процессор, поскольку PyCharm поддерживает многопоточность. Это поможет быстрее выполнить операции.
* Оперативная память: 4 ГБ свободной RAM. Исходя из опыта, лучше, чтобы общая оперативная память у устройства была не меньше 8 ГБ.
* Место на диске: 2,5 ГБ и ещё 1 ГБ для кэша. Рекомендуется использовать SSD и иметь хотя бы 5 ГБ свободного пространства на диске. Опять же, исходя из опыта, лучше иметь половина свободного дискового пространства, так как в противном случае, машина может “подтормаживать”.
* Разрешение экрана: минимальное 1024 × 768 пикселей, рекомендуется 1920 × 1080 или большее.

Версия языка Python:

• Python 2.7; в данный момент, уже используется для разработки очень редко. Обычно для поддержания существующих проектов.

• Python 3.6 или более новая версия.

Community и Pro – отличия.

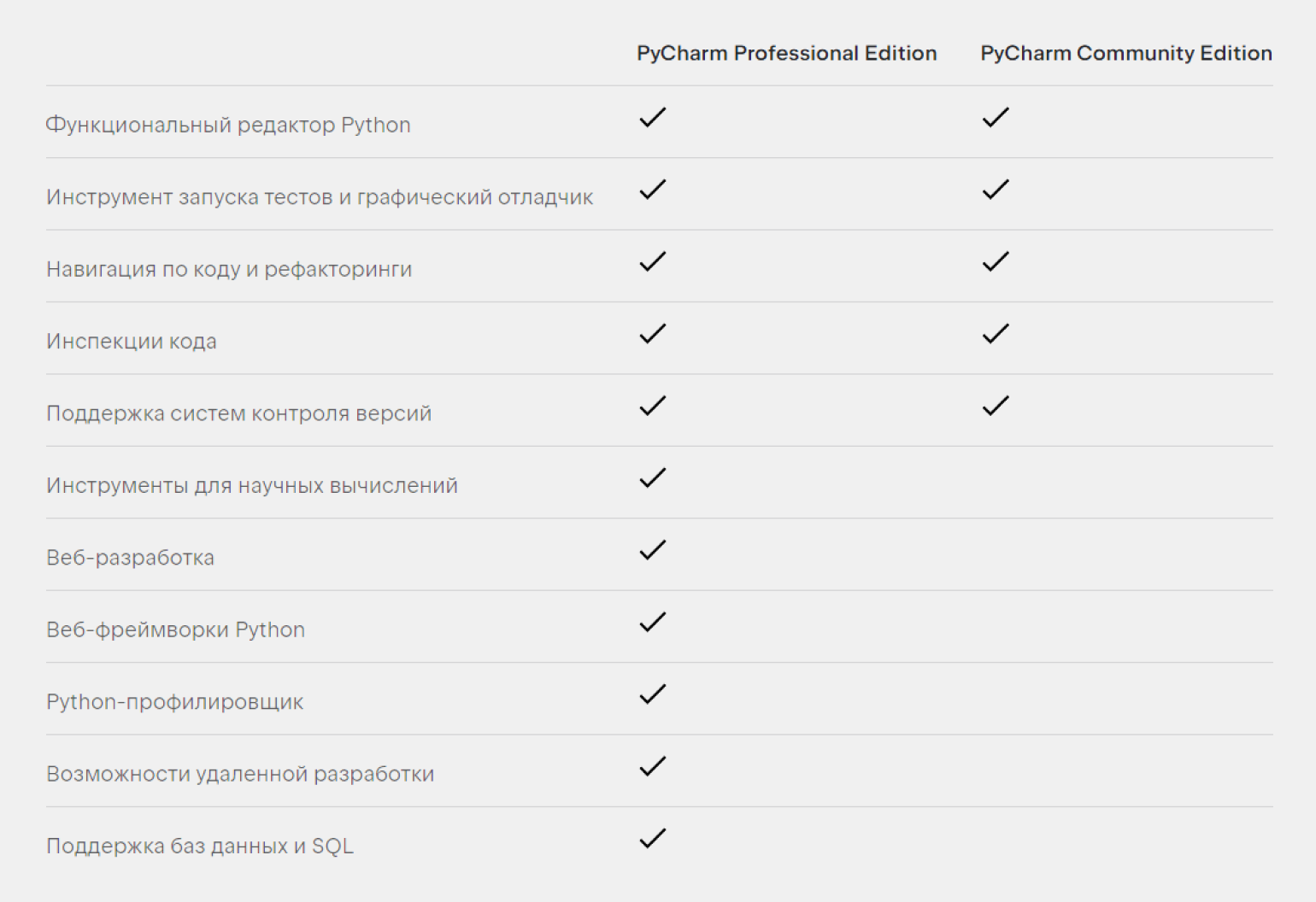
У PyCharm есть 2 варианта: первый - бесплатный называется Community и платный - Pro или Professional. Первый обычно используют для обучения или небольших проектов. Второй для профессиональной разработки крупных проектов.

**Community** предназначена для кодирования чистого Python, но поддерживает множество других языков, форматов: JSON, JYTHON, XML, XML, YAML, XSL, XPath, Markdown и другие. При помощи плагинов можно установить и Rust, и Dart.

В Professional есть всё то же, что и в Community, а так же другие функциональные способности. Например:

• backend — Cython и SQL;

• frontend — JavaScript, TypeScript, CSS, Sass, SCSS, Less. Так же доступны плагины с поддержкой Haml, Slim, Liquid.

  
Веб-разработка со всем функционалом доступна только в редакции Pro; в ней есть поддержка веб-фреймворков, таких как:

• для Python: Django, Flask, Pyramid;

• для других языков программирования: React, React Native, Angular, Node.js, Next.js, Vue.js.

В Community из фреймворков присутствует только PyQt.

Так же ряд различий:

• из инструментов сборки в Community присутствует Virtualenv/Buildout, а в Pro — дополнительно есть NPM, Webpack, Gulp, Grunt, Vite;

• в Pro есть поддержка удалённых интерпретаторов: SSH, Docker, Docker Compose, WSL, Vagrant; в Community таких возможностей нет. Придется пользоваться сторонними сервисами.

• в Community совместная разработка ограничена сеансами по 30 минут, в которых могут присутствовать не больше трёх разработчиков, в Pro ограничений по времени нет, а одновременно в сеансе может быть до 50 разработчиков.

Установка PyCharm

Скачать PyCharm можно напрямую с сайта разработчика или через использовать Toolbox — приложение для управления инструментами для разработчика, от JetBrains.

Установка через Toolbox

Для управления своим продуктом JetBrans рекомендует применять именно Toolbox. С помощью приложения можно быстро и из одного места устанавливать нужные инструменты, автоматически их обновлять сразу с плагинами, откатывать на предыдущую версию и работать параллельно с несколькими версиями.

Toolbox будет особенно удобен, если мы пользуемся сразу несколькими продуктами от JetBrains.

Toolbox для Windows

• Скачать и запустить файл в формате .exe.

• Следовать инструкциям по установке.

• Toolbox будет доступен по иконке на панели задач или рабочем столе.

Toolbox для macOS

• Скачать образ диска dmg. Обращаем внимание, что есть две версии файла: для процессоров Intel и Apple silicon.

• Смонтировать образ и перетащить JetBrains Toolbox в папку «Приложения» (Applications).

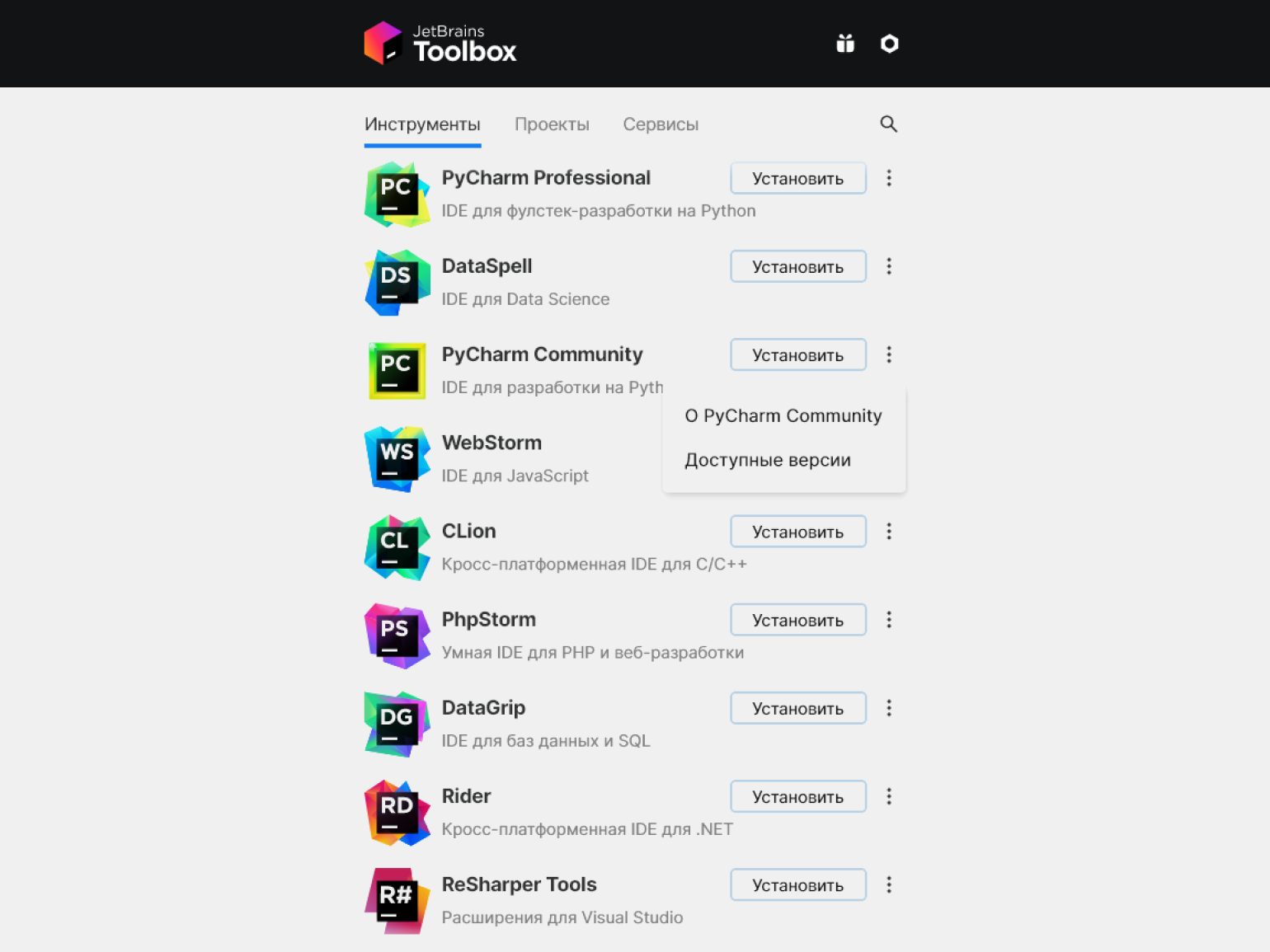
Toolbox для Linux

• Скачать архив в формате tar.gz.

• Распаковать его в директорию, которая поддерживает выполнение файлов.

• Запустить jetbrains-toolbox. После первого запуска в главном меню появится иконка, по которой Toolbox будет доступен.

После запуска Toolbox появится возможность можно скачать и запустить PyCharm прямо из приложения.



Автономная установка

Скачать PyCharm можно и отдельно, без использования стороннего приложений.

Для Windows

Скачать установку формата.exe. Есть два типа: для ARM64-процессорных компьютеров и всех других.

После скачивания файла появляется страница благодарности. Для проверки целостности установки на ней можно загружать контрольный сумм SHA.

Запустите установку, следуйте инструкциям и выберите нужные характеристики.

Для macOS

Скачать диск в формате dmpg. Их два типа: для Intel-процессоров и Apple-Silicon-процессоров.

После скачивания файла появляется страница благодарности. Для проверки целостности дискового образа можно скачать контрольный пакет SHA.

Смонтируйте в системе образ и вставьте в папку Приложения приложения PyCHarm.

Для Linux

Скачать архив формата tr.gz. Их два типа: для ARM64-процессорных компьютеров и всех других.

После скачивания файла появляется страница благодарности. Для проверки целостности архива на ней можно загружать контрольные суммы SHA.

Распакуйте скаченный файл. Если в текущей директории «Загрузки» Downloads нет поддержки выполнения файла, распакуйте файл в другой. Это можно сделать командой:

•tar xzf pycharm-\*.tar.gz -C <new\_archive\_folder>

Вместо звёздочки нужно указать версию PyCharm, вместо <new\_archive\_folder> — директорию. Производитель рекомендует использовать директорию /opt. Например:

tar xzf pycharm-community-2022.3.1.tar.gz -C /opt/

• Перейти в субдиректорию /bin

cd <new archive folder>/pycharm-\*/bin

Запустить pycharm.sh:   
sh pycharm.sh

**Bot api**

Боты являются сторонними приложениями, работающими в Telegram. Пользователи могут работать с ботами посредством отправки сообщений, команд и встраиваемых запросов. Управление ботами осуществляется через запрос HTTPS в API Telegram Bot.

Для создания простейшего робота ничего не нужно, кроме самого Telegramа. Пользователь может отправлять сообщения боту через мессенджер Telegram или API. Для взаимодействия с ботом, как владелец, ничего не нужно, кроме браузера или другой сторонней программы, которая отправит запросы HTTPS. Но это станет самым простым ботом и самым простым взаимодействием с ним.

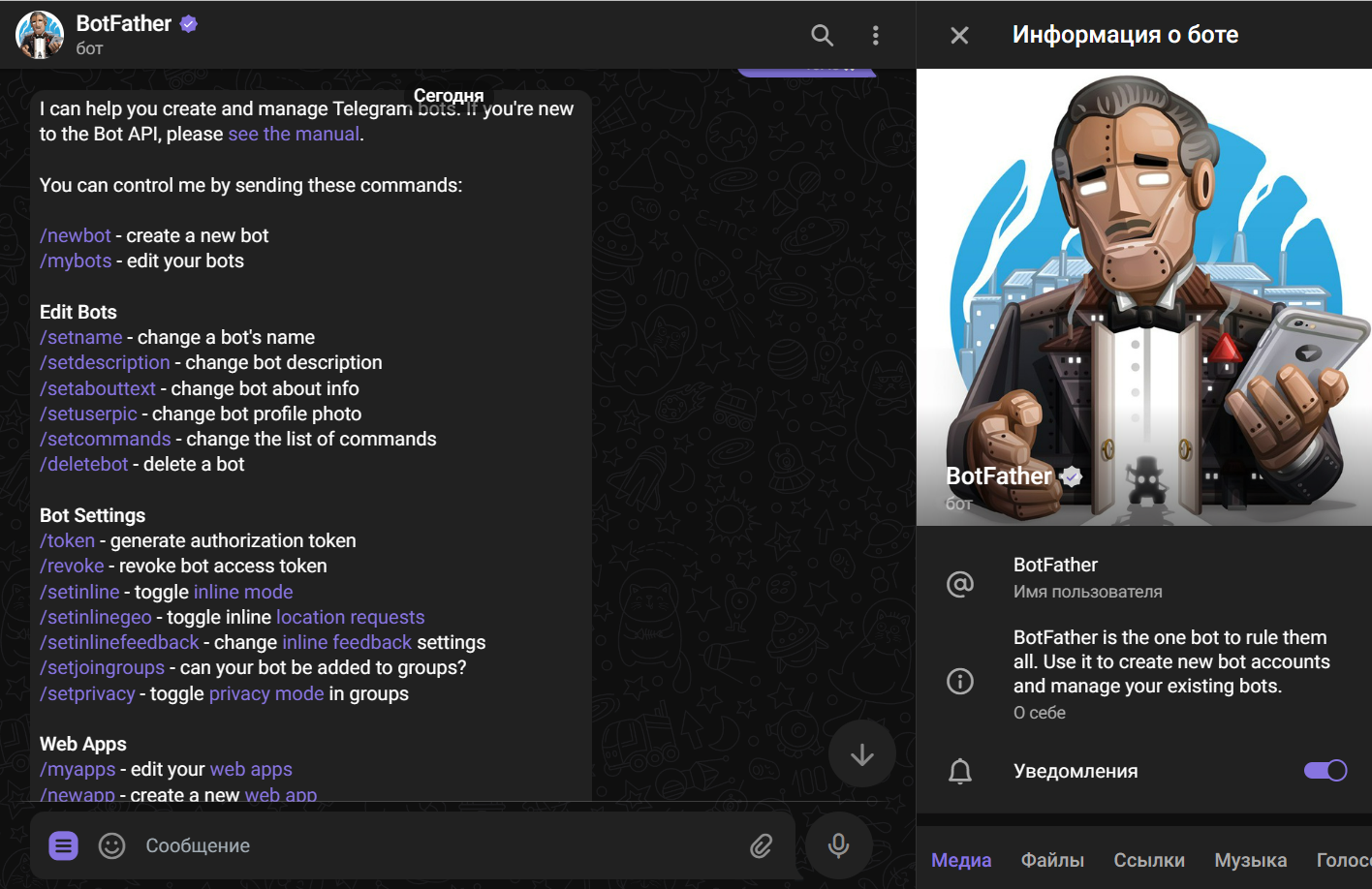
Создание (регистрация) Telegram-бота.

Боты Telegram — это определенные аккаунты, для настройки которых не требуется дополнительный телефонный номер. Нужно зарегистрировать (создать) бота, а потом пользователи смогут отправлять ему сообщения или команды, а он сможет им отвечать.

BotFather это бот, который отвечает за регистрацию и управление ботом. Это официальный бот от разработчиков Telegram. Прежде всего нужно его найти в Telegram, сообщить, что нужно зарегистрировать нового бота. После ответа на несколько простых вопросов мы получаем токен, с помощью которого мы можем взаимодействовать с ботом через API Bot, как его владелец, а пользователи могут отправить ему сообщения и команды через Telegram или API.

Находим в мессенджере BotFather (https://t.me/botfather), так же как любого другого пользователя и отправляем ему команду /newbot. Бот попросит сообщить название и имя нового бота, а взамен даст нам Token.

Мы держим полученный Токен в секрете. Все, кто знает его, могут использовать токен для контроля и управления ботом.



С данного момента можно начинать взаимодействовать с нашим ботом. Например, используя метод get, можно отправить GET запрос в Telegram Bot API и увидеть информацию о нашем боте.

https://api.telegram.org/bot {TOKEN} / {METHOD\_NAME}

где:

• TOKEN - token, который нам дал BotFather при регистрации бота.

• METHOD\_NAME - метод Bot API, который мы хотим использовать

{

"ok": true,

"result": {

"id": 976576497659,

"is\_bot": true,

"first\_name": "MyBot - тестовый бот",

"username": " MyBot \_bot",

"can\_join\_groups": true,

"can\_read\_all\_group\_messages": false,

"supports\_inline\_queries": false

}

Идентификатор бота - id. У каждого пользователя, чата или телеканала имеется такой уникальный ID. Именно такой ID применяется для взаимодействия ботов и пользователей.

В параметре is\_bot значение true говорит о том, что этот пользователь является ботом.

first\_name и username говорят за себя сами. Это мы сообщили BotFather, при регистрации нашего Бота.

"can\_join\_groups": true – говорит нам о том, что в группы можно добавить бота.

"can\_read\_all\_group\_messages": false означает, что бот не может прочитать сообщения от пользователей в группе. Это значение можно изменить на True, и бот будет читать то, что пишет ему другой пользователь.

"supports\_inline\_queries": false - говорит о том, что боту нельзя отправлять команды или сообщения, используя символ @. Можно поменять это значение на True и тогда пользователи могут отправить ему сообщение.

Чтение сообщений

После регистрации боту можно отправлять сообщения. Находим его в Telegram, нажимаем Start и отправляем определенный message.

Все сообщения отправляются на Telegram-сервер и там хранятся не более одних суток. Для получения списка сообщений существует два метода.

1.Вызвать метод getUpdates

2.Настроить webHook.

Отправка сообщений от имени бота

Самое простое средство использования Бота, которое не требует программирования или специальных программ, - отправлять сообщения от имени нашего Бота.

Например, мы можем добавить Botа в группу, а используя API Telegram Bot отправлять сообщения от имени бота в этой группе. Чтобы это сделать, необходимо использовать метод sendMessage.

В адресной строке браузера, точно так же, как это делали с методом get, вставляем URL следующего вида:

https://api.telegram.org/bot{TOKEN}>/sendMessage?chat\_id={CHAT\_ID}&text={TEXT}

где:

\* TOKEN - token, который мы получили от BotFather.

\* CHAT\_ID - ID пользователя, группы или канала, которому мы хотим отправить наше сообщение.

\* TEXT - Текст сообщения, который мы отправляем.

Одним из наглядных примеров применения данного метода мы можем назвать систему мониторинга, или любую систему, которая требует оповещения и способна отправлять запрос на HTTPS или выполнять скрипты.

Узнаем Id.

Для того, чтобы отправить кому-нибудь сообщение, необходимо знать его Id. Так же Id есть в чатах и каналах. И их id тоже надо знать для того, чтобы получить возможность отправить им сообщение.

Одним из способов выяснить id чата является добавление бота к чату и вызов метода getUpdates.

https://api.telegram.org/botTOKEN/getUpdates

Удаление прочитанных сообщений

При каждом вызове метода getUpdates мы получаем все сообщения в течение последних 24 часов. Для удаления уже обработанных файлов в методе GetUpdates необходимо передать offset c номеру обновления \_+1.

Разработчики Telegram периодически делают обновления, а сообщество давно выпускает фреймворки, которые упрощают работу в Телеграм. Некоторые из них стали очень популярными среди разработчиков:

Для Node JS (Кроссплатформенная среда исполнения с открытым исходным кодом)

Node.js Telegram Bot API

telegraf.js

Для языка Python

Python Telegram Bot

Для языка C#

.NET Client for Telegram Bot API

Установка библиотеки

Установим и настроим необходимую библиотеку Python для работы с телеграмм, например aiogram или python-telegram-bot.

Установка aiogram

Запустим командный интерпретатор (в Windows — cmd.exe)

Набераем команду:

pip install aiogram

Как установить библиотеку Python Telegram Bot в Windows, Linux и MacOS:

1. Windows:

Открываем командную строку или PowerShell.

Установим библиотеку с помощью команды: pip install python-telegram-bot

Ждем, пока процесс установки завершится.

2. Linux:

Открываем терминал.

Установим библиотеку с помощью команды: pip install python-telegram-bot.

Ждем, пока процесс установки завершится.

3. MacOS:

Откроем терминал.

Установим библиотеку с помощью команды: pip install python-telegram-bot

Ждем, пока процесс установки завершится.

После завершения установки библиотеки Python Telegram Bot можем использовать ее в своих проектах для создания Телеграм-ботов.

**Aiogram**

Aiogram - это библиотека Telegram-ботов на Python, позволяющая разработчику создавать Telegram-боты. С помощью API Aiogram можно легко и быстро создавать, настраивать и управлять ботами Telegram с помощью удобного и простого API.

Aiogram предлагает множество возможностей и функций, в том числе возможность отправки и получения сообщений, обработки команд и событий, создания и обновления диалогов, работе с файлами, мультимедиа и многом другом. Библиотека " Aiogram "

также имеет встроенный механизм обработки ошибок и логирования, что делает ее более безопасной и надежной.

Разработчики Aiogram могут создать боты с различными функциональными возможностями, например, информационными, развлекательными, маркетинговыми и другими. Например, посредством Aiogram можно создать бота для заказов, который отвечает запросам пользователей, примет заказы, будет работать с платежами, доставлять еду и многое другое. Также можно создать боты-помощники, которые будут помогать пользователям решать различные задачи и вопросы.

Поэтому Aiogram является мощным и удобным инструментом создания Telegram-ботов, позволяющим разработчикам быстрым и удобным образом создавать ботов и настраивать их работу.

Особенности Aiogram Beta 3.

* Асинхронный (документы asyncio, PEP 492)
* Присутствуют подсказки типа (PEP 484) и может использоваться с mypyramid
* Поддерживает PyPy (Интерпретатор языка программирования Python)
* Хорошо работает с Telegram Bot API 6.5 и получает быстрые обновления до последних версий Bot API
* Код интеграции Telegram Bot API генерируется автоматически и может быть легко восстановлен при обновлении AOI
* Обновления маршрутизатора (чертежи)
* Имеет конечный автомат
* Использует [мощные фильтры](https://docs.aiogram.dev/en/dev-3.x/dispatcher/filters/magic_filters.html#magic-filters)
* Промежуточные программы (входящие обновления и вызовы API)
* Дает ответы в Webhook
* Интегрированная поддержка I18n/L10n с GNU Gettext (или Fluent)

Реализация простого бота в aiogram

1. Импортируем нужные модули, классы из выбранных библиотек.

2. Создаем новый экземпляр класса бота и передаем ему полученный токен:

1. import aiogram # Импортируем модуль из выбранной библиотеки

2. bot = aiogram.Bot(token='YOUR\_BOT\_TOKEN') # Создаем экземпляр класса бота и передаем токен нашего бота

1. Создаем экземпляр класса Dispatcher для обработки всех входящих сообщений и событий:

dispatcher = aiogram.Dispatcher(bot) # Создаем экземпляр класса Dispatcher и передаем экземпляр класса нашего бота

1. Определяем обработчики для различных типов сообщений и событий:  
   @dispatcher.message\_handler() # Обработчик для обычных сообщений

async def handle\_message(message: aiogram.types.Message):

# Код для обработки сообщения

1. @dispatcher.callback\_query\_handler() # Обработчик для необходимых нам inline-кнопок

async def handle\_callback\_query(callback\_query: aiogram.types.CallbackQuery):

# Код для обработки inline-кнопки

1. Запускаем бота и начинаем обработку входящих сообщений и событий:

async def main():

await dispatcher.start\_polling()   
 # Запустить бота для получения входящих сообщений и событий.

if \_\_name\_\_ == '\_\_main\_\_':

aiogram.executor.start\_polling(dp, skip\_updates=True)   
 # Запускаем основной цикл выполнения бота

1. Запускаем программу и бот будет готов к работе.  
   import aiogram

bot = aiogram.Bot(token='YOUR\_BOT\_TOKEN')

dispatcher = aiogram.Dispatcher(bot)

@dispatcher.message\_handler()

async def handle\_message(message: aiogram.types.Message):

# Код для обработки сообщения

@dispatcher.callback\_query\_handler()

async def handle\_callback\_query(callback\_query: aiogram.types.CallbackQuery):

# Код для обработки inline-кнопки

async def main():

await dispatcher.start\_polling()

if \_\_name\_\_ == '\_\_main\_\_':

aiogram.executor.start\_polling(dp, skip\_updates=True)

Пример простого бота на aiogram

import logging

from aiogram import Bot, Dispatcher, types

from aiogram.contrib.fsm\_storage.memory import MemoryStorage

# Установка уровня логов для отладки

logging.basicConfig(level=logging.INFO)

# Инициализация бота и диспетчера

bot = Bot(token="BOT\_TOKEN")

storage = MemoryStorage()

dp = Dispatcher(bot, storage=storage)

# Обработчик команды /start

@dp.message\_handler(commands=['start'])

async def start\_command(message: types.Message):

# Отправка приветственного сообщения

await message.answer("Привет! Я бот, который может тебе помочь.")

# Создание кнопки с нужной нам ссылкой

button = types.InlineKeyboardButton(text="Перейти на сайт", url="https://ссылка на необходимый сайт")

keyboard = types.InlineKeyboardMarkup().add(button)

# Отправка кнопки пользователю

await message.answer("Нажми кнопку, чтобы перейти по ссылке:", reply\_markup=keyboard)

# Запуск бота

if \_\_name\_\_ == '\_\_main\_\_':

from aiogram import executor

executor.start\_polling(dp, skip\_updates=True)Python

После этого можно запустить бота, и он будет приветствовать пользователей и создаст кнопку, которая ссылается на указанный нами сайт.

**Beautiful Soup**

Beautiful Soup является библиотекой Python для обработки HTML и XML документов. Очень часто ее используют для сканирования веб-сайтов. BeautifulSoup превращает сложные HTML-документы в сложные деревья Python объектов, например тегов, навигационных строк или комментариев.

Библиотека Beautiful Soup4 предлагает одно интерфейсное решение для разных парсеров и типов данных, но не все парсеры одинаковые. В разных парсерах, анализирующих один и тот же документ, создаются различные HTML-деревья.

Самые крупные различия будут в парсерах HTML и XML. Также парсеры отличаются скоростью обработки HTML-документа.

Если дать Beautiful Soup4 идеально оформленный HTML-документ, то различий в построенном HTML-дереве не будет, но, как правило идеально оформленные HTML-документы встречаются очень редко. Один парсер быстрее другого, но они все дадут похожую структуру, выглядящую как оригинальный HTML-документ. Но, если документ оформлен с ошибкой или какими-то особенностями, что встречается чаще всего, то разные парсеры дадут и разный результат.

Различия в построении HTML-дерева разными парсерами, покажу на короткой HTML-разметке: <a> </p>.

Парсер lxml.

Характеристики:

• Для запуска примера, необходимо установить модуль lxml.

• Очень быстрый и имеет внешнюю зависимость от языка C.

• Не строгий.

>>> from bs4 import BeautifulSoup //Импортируем библиотеку

>>> BeautifulSoup("<a> </p>", "lxml")

# <html><body><a></a></body></html>

Тег <a> заключен в теги <body> и <html>, а висячий тег </p> просто игнорируется.

Парсер html5lib.

Характеристики:

• Для запуска примера, необходимо установить модуль html5lib.

• Очень медленный.

• Разбирает страницы так же, как это делает браузер, создавая валидный HTML5.

>>> from bs4 import BeautifulSoup //Импортируем библиотеку

>>> BeautifulSoup("<a></p>", "html5lib")

# <html><head></head><body><a><p></p></a></body></html>

Парсер не игнорирует висячий тег </p>, и к тому же добавляет открывающий тег <p>.html5lib добавляет пустой тег <head> (lxml этого не делает).

Встроенный в Python парсер html. parser.

Характеристики:

• Не требует дополнительной установки.

• Хорошая скорость, но не такой быстрый, как lxml.

• Более строгий, чем html5lib.

>>> from bs4 import BeautifulSoup //Импортируем библиотеку

>>> BeautifulSoup("<a></p>", 'html.parser')

# <a></a>

Как и встроенный парсер Python не обращает внимания на закрывающий тег **</p>.**  Этот парсер, в отличие от HTML5lib, не пытается создать идеальный HTML-документ с добавлением тегов html и body.

Вывод: HTML5lib - парсер использует методы, являющиеся частью HTML5 стандарта, поэтому можем полагать, что подход его самый «правильный».

Основные правила работы с BeautifulSoup4.

Для того, чтобы разобрать HTML-документ, необходимо передать его в конструктор класса BeautifulSoup(). Возможно передать или строку или открытый дескриптор файла:

from bs4 import BeautifulSoup //Импортируем библиотеку

# передаем объект интересующего нас файла.

with open("index.html") as fp:

soup = BeautifulSoup(fp, 'html.parser')

# передаем строку.

soup = BeautifulSoup ("<html>a web page</html>", 'html.parser')

Сначала документ конвертируется в Unicode (Стандарт кодирования символов), а HTML-мнемоники конвертируются в символы Unicode:

>>> from bs4 import BeautifulSoup

>>> html = "<html><head></head><body>Sacr&eacute; bleu!</body></html>"

>>> parse = BeautifulSoup(html, 'html.parser')

>>> print(parse)

# <html><head></head><body>Sacré bleu!</body></html>

**3. Практическая часть  
Разработка алгоритма парсинга**Парсинг сайтов на Python.

Исходные данные являются фундаментом успешной работы в области анализа и обработки информации. Источников данных множество, и сайты - один из таких. Часто это может быть вторичный источник информации, такие как: сайты Worldometers, сайты новостей CNBC, соцсети, электронные коммерческие платформы, и т.д. Эти сайты предоставляют необходимую информацию для анализа и обработки данных.

Решением задачи сбора будет парсинг сайтов в Python. Для этого языка программирования имеется мощная библиотека BeautifulSoup, описанная выше, и инструмент автоматизации Selenium. Оба их часто используют специалисты для сбора данных различных форматов. В данном разделе мы начнем знакомиться сначала с BeautifulSoup.

Парсинг (англ. Parsing) - разбор, анализ или веб-скрейпинг - автоматизированный сбор данных с интернет-ресурсов. Например, мы можете собирать статьи с названиями с любых сайтов, что будет полезно в журналистике и блогерам. Программы для сбора и обработки информации из Сети называются парсерами.

Сам парсинг применяется для решений различных задач: телеграм-боты могут получить информацию и затем показывать ее пользователям, маркетинговые боты могут подтянуть данные из соцсетей, бизнесмены могут узнать подробности конкурентов.

Существуют разные подходы для парсинга: мы можем забрать информацию с помощью API, предусмотренного создателями сервиса или получить ее прямо из HTML кода. В любом случае важно помнить, как мы вообще взаимодействуем с серверами в Интернете, как работают запросы HTTP.

HTTP-запросы, XML и JSON

HTTP (HyperText Transfer Protocol, протокол передачи гипертекста) — протокол для передачи различных форматов данных между клиентом и сервером. Он называется так, потому что изначально использовался для обмена гипертекстовыми документами в формате HTML.

Для понимания работы HTTP необходимо помнить о том, что это структура клиент-сервер. Мы хотим получить какую-то информацию в интернете. Браузер, формирует запросы и отправляет их на сервер, на сервере запросы обрабатываются, формируется ответ (response) и передаётся обратно клиенту. В нашем примере клиент — это браузер. Браузер обрабатывает полученный ответ и выводит его на экран в виде страницы сайта.

Запрос состоит из трёх частей:

* Строка запроса (request line): указывается метод, который используется для передачи, версия используемого HTTP и сам URL, к которому мы делаем запрос.
* Заголовок (message header): само сообщение, передаваемое серверу, его параметры и дополнительная информация о нашем запросе.
* Тело сообщения (entity body): данные, передаваемые в запросе. Это необязательная часть, но чаще всего она присутствует.

Простой HTTP запрос, который мы используем для того, чтобы получить, к примеру, новости.

\_GET /https://dzen.ru/news?issue\_tld=ru / HTTP/1.1\_

В этом запросе можно выделить три части:

• \_GET — метод запроса. Метод GET позволяет получить данные с ресурса, не изменяя их.

• https://dzen.ru/news?issue\_tld=ru— URL сайта, к которому мы обращаемся.

• HTTP/1.1\_ — указание на версию HTTP.

Также ответ на запрос имеет 3 части: \_HTTP/1.1 200 OK\_. В начале указаны версии HTTP, цифровые ответы и текстовые пояснения. Есть десятки ответов. Сам запрос HTTP можно написать в разных форматах. Мы рассмотрим два наиболее популярных: JSON и XML.

JSON на английском языке. JavaScript Object Notation - это простой формат обмена информацией, созданный с помощью JavaScript. В этом случае используется текст, который легко воспринимается человеком, что позволяет его легко понимать и писать:

({

<font color="#069">"firstName"</font> : <font color="#069">"Валерий"</font>,

<font color="#069">"lastName"</font> : <font color="#069">"Таганов"</font>

});

Для того, чтобы нам получить информацию в формате JSON, необходимо подготовить правильный HTTP-запрос:

var request = new XMLHttpRequest();

request.open('GET', requestURL);

request.responseType = 'document';

request.send();

JSON легче для человеческого восприятия. Но выбор между определённым форматом HTTP-запроса зависит от решаемой нами задачи.

Разберемся с основами веб-скрейпинга - используем стандартные Python библиотеки и поработаем с различными полезными средствами.

Три этапа парсинга

Самое простое разобраться с парсингом - спарсить что-то. Создаем программу для показа информации о товарах.

Чтобы это сделать, пройдём три последовательных шага:

* Подключаем библиотеки, помогающие спарсить информацию при помощи Python.
* Заходим на сайт, с которого мы планируем парсить информацию и изучаем исходные данные. Важно найти элементы с нужной информацией.
* Пишем код и парсим.

Шаг 1.

Подключаем библиотеки.

Разные языки программирования имеют свою библиотеку для копирования информации с сайта. Например, JavaScript использует библиотеку Puppeteer, а Python популярна библиотека «Bs4». Их принципы работы похожи. Сначала надо разобраться, как запустить Python на ПК.

Просто так не получится написать код в текстовом документе. Можно использовать один из способов:

* Использовать терминал на MacOS или Linux или использовать командную строку Windows. Чтобы это сделать, нужно предварительно установить в систему Python.
* Воспользоваться приложением pyChamm или каким-то другим Python-приложением.

После того, как мы установили Python на компьютер и запустили pyСharm, переходим на импорт библиотек.

BeautifulSoup - это библиотека, позволяющая работать с кодами HTML и XML. Ее очень просто подключить:

from bs import BeautifulSoup

Дополнительно потребуется библиотека requests, которая поможет сделать запрос на нужный нам адрес сайта. Импортируем:

import requests

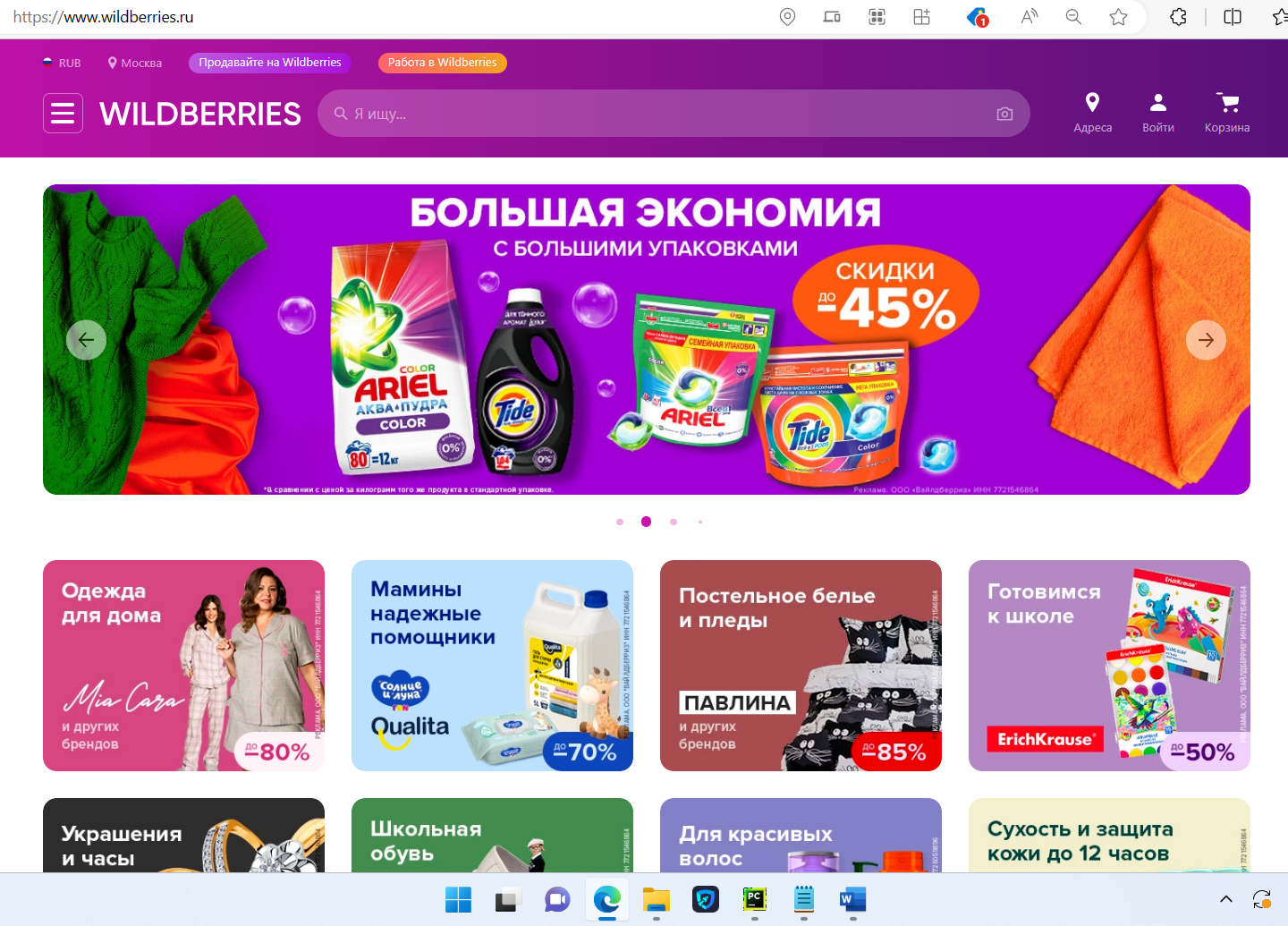
Всё. Все необходимые библиотеки готовы к работе — они помогут нам получить исходный код сайта и найти в нём интересующую нас информацию.

Библиотека Beautiful Soup обычно предустановлена в используемой среде разработки или в Jupyter Notebook, но иногда ее нет. Если при попытке её импорта мы получаем ошибку, то выполним команду для её установки, а потом повторим запросы на импорт:

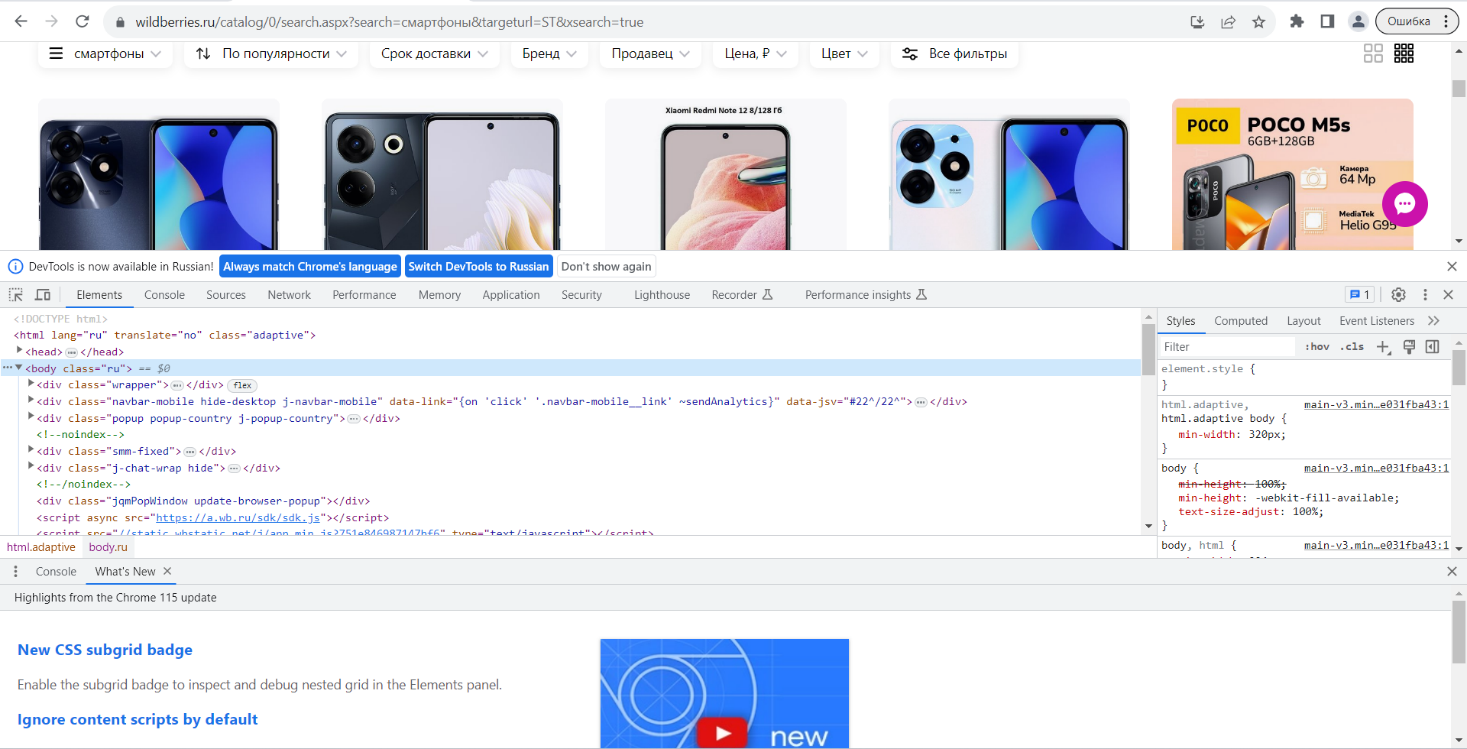
pip3 install bs4

**Шаг 2**

**Изучаем исходный код сайта**

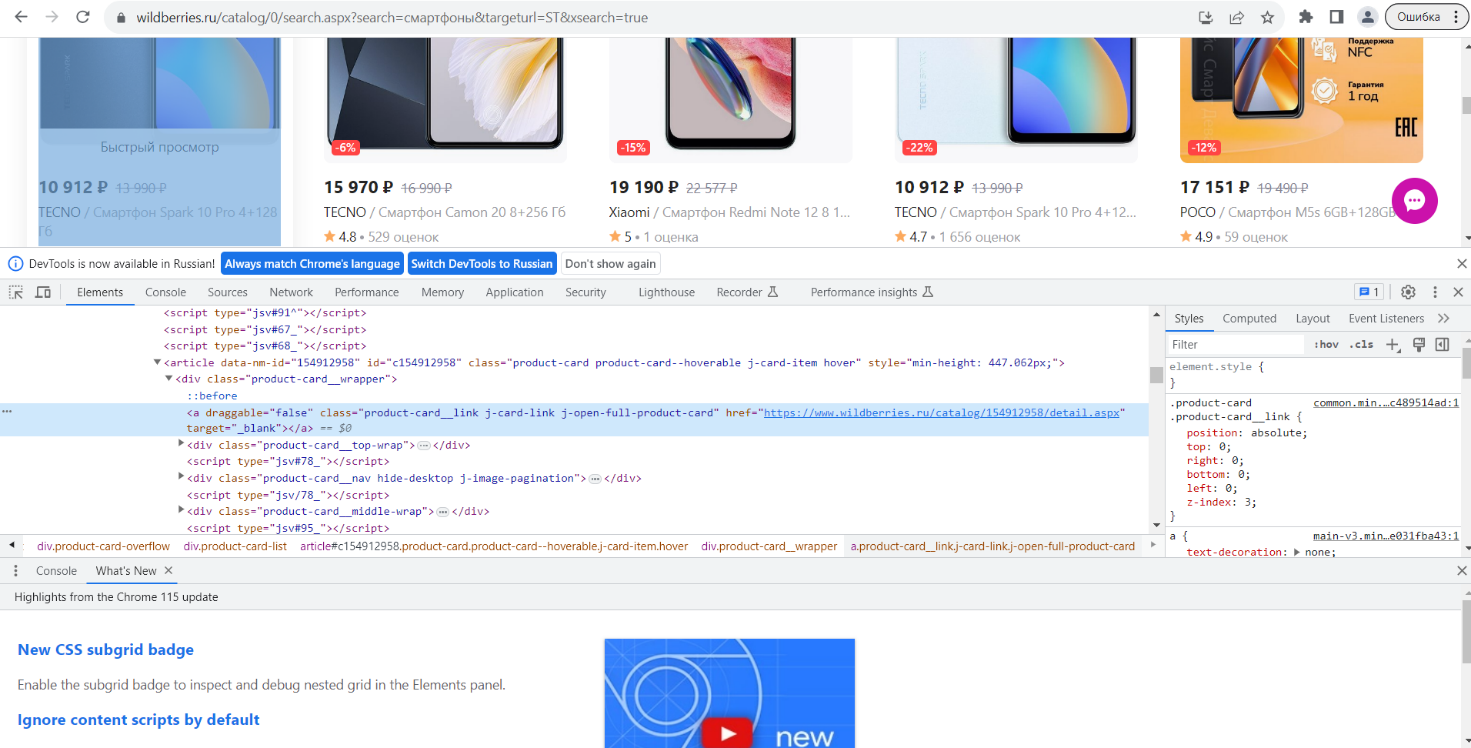
В качестве источника будем использовать [Wildberries – Интернет-магазин модной одежды и обуви](https://www.wildberries.ru/). 

Теперь посмотрим на исходный код страницы и найдём место, где хранится текущая информация. Нас интересует страница с карточками товаров:

Для просмотра HTML-кода откроем «Инспектор кода». Для этого можно использовать комбинации горячих клавиш: в Google Chrome на macOS — ⌥ + ⌘ + I, на Windows — Сtrl + Shift + I или F12. Инспектор кода выглядит как дополнительное окно в браузере с несколькими вкладками: 

Переключаться между вкладками не надо, так как вся необходимая информация уже есть на первой.

Теперь найдём блок в коде, где хранятся нужные нам данные. Для этого следует последовательно разворачивать блоки кода, располагающиеся внутри тега <body>.

Инспектор кода при наведении на блок кода подсвечивает на сайте ту область, за которую он отвечает. Переходим последовательно вглубь HTML-кода и находим нужный нам элемент.  
  
  
 **Шаг 3  
Пишем код и получаем необходимую информацию**

Продолжаем записывать команды в терминале, командной строке, IDE или онлайн Python-редакторе. На этом этапе нам остается использовать подключенные библиотеки, а также выбрать значения из объекта. Сначала нужно проверить работу библиотек.

Сохраняем в переменной URL страницу, с которой планируем брать информацию:

url=https://www.wildberries.ru/catalog/0/search.aspx?search=%D1%81%D0%BC%D0%B0%D1%80%D1%82%D1%84%D0%BE%D0%BD%D1%8B&targeturl=ST&xsearch=true

Создадим к ней запрос и посмотрим, что вернёт сервер:

response = requests.get(url)

print(response)

В нашем случае получаем ответ:

<Response [200]>

Хорошо. Ответ 200 означает, что библиотека запросов работает корректно, а сервер отдает нам информацию со страницы.

Теперь получим исходник, используя библиотеку BeautifulSoup и сразу выведем полученный результат на экран:

bs = BeautifulSoup(response.text,"lxml")

print(bs)

После этого на экране полностью виден код необходимой нам страницы.

Нам весь код не нужен – необходимо вывести только интересующий нас блок кодов, в котором хранится информация о товарах. Откроем его, используя функцию библиотеки BS.

Функция Find принимает два аргумента:

* Указание на тип HTML-элемента, где осуществляется поиск;
* Наименование этого элемента.

Мы смогли получить данные по товару с помощью библиотеки Beautiful Soup в Python. Можно использовать данную библиотеку для своих задач, например, передачи в виджет на собственном сайте, или создание нашего бота для парсинга.

Парсинг динамических сайтов с помощью Python и библиотеки Selenium.

В бесплатной библиотеке Selenium можно эмулировать веб-браузер - т.е. «маскировать» скрипты веб-запроса под действие пользователя в Google Chrome или Safari. Сайты могут распознавать боты и блокировать IP-адреса, с которых отправляются автоматические запросы.

Для того, чтобы избежать бан, можно использовать два способа: изучение HTTP, принципов работы Python в вебе. После чего написать свой эмулятор с нуля или воспользоваться готовым инструментом.   
Во-втором варианте Selenium является одним из самых удачных и удобных решений.

Задержка по времени

Не забываем про задержку по времени. Иначе сайт заблокирует наш адрес.

Чтобы сделать задержку в парсере, можно использовать следующий код на Python:

import time //Импортируем библиотеку

def sleep\_for(seconds):

time.sleep(seconds)

sleep\_for(5) # задержка на 5 секунд

Здесь мы используем функцию time.sleep(), чтобы выполнить задержку. Мы передаем ей количество секунд, которое хотим подождать, и функция выполняет задержку перед выполнением кода.

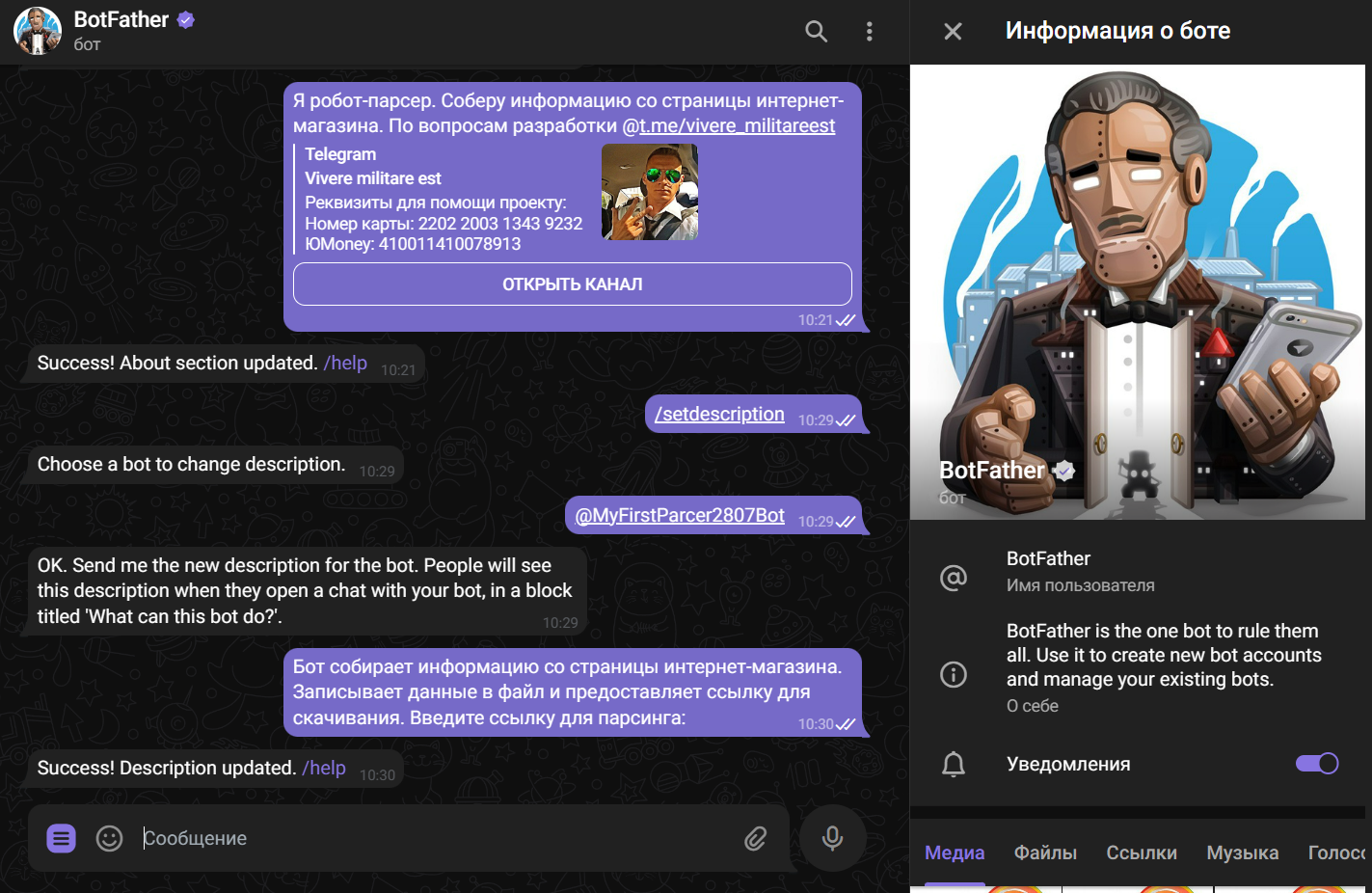
Можно использовать эту функцию в своем парсере для задержки между запросами к серверу или между действиями внутри парсера.

Парсинг дает возможность получить необходимую информацию со всех сайтов. Для него могут использоваться различные языки программирования, но в некоторых из них есть стандартные библиотеки для web-скрейпинга, например Beautiful Soup в Python. Парсинг помогает получить нужную нам информацию с любого сайта. Для него можно использовать различные языки программирования, но некоторые из них уже содержат стандартные библиотеки для веб-скрейпинга, например Beautiful Soup на Python.

**Создание Telegram-бота**

### Шаг 1. Регистрация бота в BotFather

В первую очередь зарегистрируем бота в Telegram.



Пошаговая инструкция описана в теоретической части.  
  
Шаг 2. Подготовка необходимых компонентов

Подготовим нужные компоненты. На компьютере мы уже установили Python, а среда для разработки в нашем случае - PyCharm.

В качестве библиотеки мы будем пользоваться pyTelegramBotAPI. Установить ее можно при помощи следующей команды:

pip install pyTelegramBotAPI

Все необходимые комплектующие установлены в нужном порядке. Перейдем к следующему шагу.

Шаг 3. Написание кода

Во-первых, мы подключаем установленную нами библиотеку и используем уникальный токен, о котором мы говорили выше, а во-вторых, импортируем типы, которые понадобятся для работы кнопок.

botTimeWeb = telebot.TeleBot('Уникальный токен')

from telebot import types

Вместо ‘Уникальный токен’ указываем токен, полученный при регистрации бота.

Реализуем обработчики сообщения. Они принимают участие в фильтрации входящих сообщений Бота и вызывают указанную нами функцию.

Добавлен метод отправки сообщений send\_message, необходимый для того, чтобы отправить пользователю сообщение. Он имеет 4 параметра:

• message.chat.id отвечает за синхронизацию сообщения с чатом бота по id;

• first\_mess передает первое сообщение;

• parse\_mode необходим для указания режима разметки в сообщениях нашего бота;

• reply\_markup отвечает за добавление созданной нами кнопки.

Бот создан. Сохраним файл с кодом и перейдем к запуску.

Шаг 4. Запуск бота

Откроем терминал и выполним запуск проекта:

python main.py

Теперь бот запущен, можно перейти в Telegram и тестировать работу нашего бота.

**Тестирование Telegram-бота**

В сети много информации о создании чат-ботов, но очень мало информации о тестировании. Свойства и специфика тестирования современного чат-бота рассматриваются на примере платформы Telegram, которая является наиболее популярным веб-продуктом в этой области.

Плюсы и минусы чат-ботов

Чтобы проанализировать это, обратимся к понятию «чат-бот». Отмечу, что существует 2 вида ботов.

* Чат-боты с фиксированными данными в BD. Их работа предназначена для ограниченного спектра сценариев поведения, таких как бот-помощник и кнопочный бот;
* Чат бот на основе ИИ, с параметрами обучения и обновления информации, например, «умный Бот».

Также необходимо заметить, что чат-боты с фиксированными данными надежнее в работе, поскольку не содержат ИИ-наработки, т.е. не могут быть выведены из-под контроля программистами. А вот роботы с ИИ становятся более «отзывчивыми», а их интеллект представляет собой технические возможности для реагирования на текущую ситуацию.

Теперь можем выделить плюсы и минусы для чат-ботов после их тестирования в условиях среднестатистической компании.

Преимущества:

1. Круглосуточная работа;

2. Быстрый ответ;

3. Простой и понятный графический интерфейс;

4. Постоянное расширение покрытия платформ;

5. Для взаимодействия с чат-ботом не нужно устанавливать стороннего ПО.

Недостатки:

1. Множество функций;

2. Иногда нестандартное поведение пользователя может вызвать сбой в работе бота;

3. Подходит не для всех типов бизнеса.

Тестирование чат-ботов.

Вначале, стоит ознакомиться с тем, зачем создается чат-бот, узнать потенциальную аудиторию, а также решить, какие задачи и операции он должен выполнять для клиентов. Например, у руководителя проекта можно спросить, какие команды будут необходимы боту для взаимодействия с чатом, могут. Возможно, список подобных команд будет значительно разнится.

Также нужно определить, на какие конкретные страны будет ориентирован наш чат-бот, а также какие языки в нем будут поддерживаться. И дополнительно можно уточнить на каких операционных платформах будет развиваться бот.

На начальном тестирования необходимо пройти базовые позитивные сценарии и проанализировать насколько интуитивно понятно взаимодействие с нашим чат-ботом. Тестовые диалоги с ботами не должны содержать тупики — любое действие пользователя должно находить свой отклик со стороны чат-бота.

Если Бот поддерживает различные локализации, необходимо проверить наличие правильного перевода для потенциальных диалогов с Ботом и логичность их построения. При этом следует учитывать культурные и исторические особенности государства, языковые и орфографические ньюансы формирования слов.

Дополнительно необходимо учитывать моменты регистрации и валидацию номеров телефона и адреса, соответствуют ли они формату выбранной нами страны.

И, наконец, стоит убедиться, что элементы и объекты, которые обозначают одно и то же, именуются одинаково. При удалении выбранного элемента должно отображаться соответствующее сообщение о подтверждении действия. Протестировать текущее соответствие завалидированному дизайну (макету) — отлично, если графическое отображение чат-бота выдержано в единой стилистике и цветовой гамме для разных операционных систем.

Итоги.

Естественно, не получится проверить все сценарии, но проверить бота на базе утвержденных тестовых документов, выполнить все оговоренные клиентом положительные сценарии, провести исследовательские испытания - прямая обязанность QA инженера.

Подход для тестирования ботов должен быть аналогичен проверке веб приложения, хотя все области имеют свои особенности, а также похожие моменты.

**Оценка эффективности работы бота**

Чат-бот является инструментом автоматизированного общения с клиентом. С помощью бота можно повысить воронку продаж и улучшить взаимодействие с клиентами. Но в этом случае бот может и ухудшить конвертацию, если он будет выполнен некорректно. Разберемся, как работать с чат-ботом и какие параметры стоит учитывать в анализе.

Метрики для оценки чат-бота

Основными характеристиками успешного чат-бота являются минимальные отписки, растущая аудитория, соответствующий контент, высокий процент вовлеченности.

Рост активной аудитории

Если у бота стабильно будет больше отписок, чем подписок, то такой бот скорее всего просто прекратит существование. В инструменте должна быть польза для клиента, содержать соответствующие материалы.

Качество обработки бот-запросов влияет на рост аудитории. В чат-боте должны быть выполнены заявленные функции. Перед запуском бота в работу проверяем, насколько он понятен и прост в применении. Основными задачами, которые должна решать бот, являются информирование клиентов, прямое целенаправленное действие оставления контактов, оформления заявки, перехода на ленд и преленд, а также по необходимости обращение к менеджеру колл центра.

Нужно запомнить, что перезагрузка бота функционалом плохая идея. Выделяем основную задачу бота, дополняя ее 2-3 функциями, но не больше. Путь к цели должен быть прост.

Часто основная ошибка создания бота – желание всё реализовать и сделать сразу все. Потому что человек быстро принимает решение. Бота нужно настроить так, чтобы бот не "сбивал" с цели.

Например, бот для мониторинга за скидками на Wildberries. Все, что необходимо сделать - отправить Боту ссылку на интересующие продукты. Как только цены на товар изменятся, Бот сообщит об этом и клиент может сделать интересующий его заказ. Существует скидка – и целевое действие. Товары могут быть удалены и добавлены из списка отслеживания. У бота свой реферал – чем больше вы приведете людей, тем больше товара на Wildberries вы можете отслеживать.

Органический прирост

Приток людей должен быть непрерывным, не только во время действия рекламных кампаний или маркетинговых мероприятий. Чтобы пользователи пришли к нам, мы размещаем данные о боте во всех возможных местах связи с пользователями: социальные сети, сайты, лендинги, добавляем к физическим носителям QR-код или загружаем посты в паблики. Продвигать бота можно также с помощью лид-магнита, сообщая о том, что, заказывая через Бота, пользователи получают дополнительные скидки, подарки и другие выгоды. Это поможет нарастить количество органического трафика в боте.

Название чата влияет на поток пользователей. Оно должно быть простое, чтобы его можно было легко запомнить и найти в поиске в Telegram. Сложное название всегда плохо работает, чем меньше символов надо вбить потенциальному покупателю, тем лучше. Хорошо если названия совпадают с товаром или услугой. Это помогает пользователю лучше ориентироваться, при выборе виртуального собеседника.

Пример оформления бота

Уровень вовлеченности и низкий процент отписок

Важно анализировать, как реагируют подписчики на рассылки. Средняя цифра, на которую нужно ориентироваться, - не менее десяти процентов активных посетителей. Если создать чат-бот при помощи конструктора, то он уже имеет встроенную аналитику, позволяющую следить за вовлеченностью в контент. Также, если добавить в бот Tgstat, мы сможем анализировать его через сервис.

Смотрим, наиболее интересные темы и после каких сообщений активнее отписываются. Это помогает оптимизировать работу бота и подобрать оптимальную тему и время сообщения. Даем соответствующий контент – именно поэтому пользователи приходят в чат. Информация относится только к нашему продукту или услуге. Хорошо если это будет разные бонусы и скидки, для того чтобы клиент увидел выгоду, и мы смогли его «подогреть» своей рассылкой. Изучая показатели, мы можем найти свою наибольшую ценность и определить, как нам лучше взаимодействовать со своей ЦА.

В общем, для анализа ботов используются три категории KPI:

* Информацию о том, как выполняются ботом бизнес-задачи, для которых он был создан;
* Метрики, которые отражают востребованность бота;
* Метрики, которые помогают определить, насколько эффективен диалог (Общение с пользователем).

· Многие из метрик чат-ботов также используются в обычных интернет-маркетинге. Впрочем, их интерпретация в диалоговой коммерции часто отличается от привычной. Кроме того, боты создаются с различными целями, а метрика, которая важна для того или иного типа диалога, может не означать ничего для другого чат-бота.

Решение бизнес-задач.

Этот набор метрик является самым главным, поскольку помогает определить срок окупаемости затрат на разработку нашего бота и соответствует ли он вообще нашим ожиданиям. Сам KPI в этом случае может быть абсолютно разным и зависеть только от функций бота. Приведу несколько наиболее популярных примеров:

Снизить загрузку колл-центра. Эта задача является одной из наиболее распространенных в чат-ботах, в данное время. Поскольку большая часть запросов в Колл-центре стандартная, боты с ними справляются не хуже, чем люди. Осуществляют обслуживание пользователей в чате и на телефоне, переводят на «живых» операторов, только при определенной необходимости. Внедрение бота позволяет сократить загрузку сотрудников колл-центра на 70 процентов

Конверсии в покупки. Эта простая метрика актуальная для интернет-магазина и применяется также в традиционной маркетинговой деятельности. Бренды, такие как, к примеру H&M, используют бота как интернет-магазин в мессенджере Телеграм. Если это наш случай, то продажи для нас - главная метрика. Иногда боты умудряются добиться уникального результата. Увеличиваются продажи – увеличивается наша прибыль. Например, интернет-магазин Asus смог с его помощью увеличить конверсию в покупки на 300 процентов.

Активация. Одним из плюсов ботов является то, что после того, как они добавляются в друзья к пользователю, они периодически могут напоминать о себе и нашем бизнесе. Например, информируя о новинках в магазине и акциях, предлагая интересную информацию о бренде. Можно снабжать подобные сообщения ссылками и таргетировать их на пользователей, которые не появлялись на нашем сайте уже давно. А в итоге - рассчитывать процент пользователей, которых мы «вернули» на наш сайт.

Рост корпоративного CRM. Многие кейсы с использования ботов направлены прежде всего на развитие CRM. Посетители, которые хотят поиграться с ботом, вступают в коммуникацию с компанией. Для того чтобы понять, как бот помогает расширить базу клиентов, нужно следить за тем, для кого из пользователей чат стал первым пунктом контакта с брендом нашей компании. Например, компания flowers проведя расчет, увидела, что 80 процентов заявок, поступающих через Бота, являются заказами от новых пользователей.

Рост числа упоминаний бренда. Не секрет, что часто одна из главных причин стартапа – желание получать дополнительный PR бонус при использовании новых технологий. Интересные кейсы, в которых бот с пользователями играет или предлагает необыкновенный сервис - прекрасный информационный повод.

Для отслеживания этой метрики необходимо следить за информацией о нашем боте в СМИ или блогах, а также за количеством пользователей, которые поделились своим ботом в соцсетях. Кстати, для разработки бота стоит обеспечить его социальными инструментами.

Если какие-то из ключевых метрик для нашей компании не выглядит оптимистичными, то это повод для обращения к двум другим метрикам и с помощью них можно понять, почему система не работает так, как мы хотели бы.

Востребованность чат-бота.

Количество посетителей. По сути, абсолютное количество пользователей бота говорит не только о своей эффективности, но и о мощности наших каналов, посещаемости сайта, продуманности инструментов, благодаря которым мы добавляем бота в друзья к пользователям. Впрочем, существуют некоторые дополнительные метрики, помогающие оценить, насколько бот хорошо работает:

* Вовлеченность пользователей. Вовлеченный пользователь — это пользователь, который хоть раз вступил в контакт с ботом, т.е. ответил на сообщение или написал свое. Количество вовлеченных пользователей, измеряется процентами от общего числа пользователей, добавленных в друзья к боту. Хорошим способом оценить активность бота является сравнение ее с активностью других каналов, например, почтовых рассылок или рекламой в соцсетях. Если бот является точкой входа на все обращения пользователей, например колл - центр этот параметр не актуален.
* Активные пользователи. В этом проценте пользователей, которые написали боту самостоятельно, а не ответили на его очередной пост. Эта метрика отражает одну из основных признаков успешности бота – его полезности. Если люди сами пишут боту, то сервисы, предлагаемые системой, важны для тех, кто пишет нашему боту. Эта метрика также связана с желанием пользователей советовать бота друзьями.
* Повторные пользователи. Две предыдущих метрики могут привести к высокому результату просто потому, что людям интересно взаимодействовать с новыми технологиями. Впрочем, часто первая связь с ботом становится последней. Иногда достаточно, к примеру, если бот используется для увеличения известности бренда, или для сбора данных пользователей. Но если мы ожидали регулярного использования системы, то низкие показатели или падение этой метрики — повод пересмотреть сценарии взаимодействия.

Количество прочитанных сообщений. Особенно актуальна эта метрика в том случае, если мы используем бота для передачи веерной рассылки. К примеру, для информирования клиентов интернет-магазина о скидках и акциях. Важно помнить, что в настоящее время в диалоговой сфере количество читаемых сообщений, как правило, превышает такие же показатели на других каналах. В среднем читаемость составляет 40-70 процентов.

Эффективность диалогов

Средняя длительность сеанса. В зависимости от цели, для которой мы создали бота, эта метрика может быть прочитана совершенно противоположным образом. Одним из основных преимуществ чат-ботов является то, что они могут быстро справляться с большими задачами, быстрее, чем живые операторы и сами пользователи. Для таких ситуаций короткая беседа является хорошим знаком при условии выполнения задачи.

В остальных случаях разработчики надеются на то, чтобы пользователи как можно дольше общались с их ботом. Обычно это ситуации, когда бот занимается развлечениями пользователей, исполняет роль консультанта и друга. Такие боты чаще используют компании для того, чтобы повысить узнаваемость бренда, создать эмоциональную связь бренда и пользователей. Длинная беседа также хорошо подходит для медиа-ботов, цель которых заключается в том, чтобы вовлечь пользователей в общение с содержанием.

Процент ошибок. Чат-боты используют технологию NLP (Natural Language Processing) и могут понять сообщения обычного, естественного человеческого языка. Впрочем, ни один бот в мире до сих пор не достиг стопроцентного успеха в NLP.

Боты зачастую неправильно понимают людей или вообще не понимают их. Ничего страшного в этом нет, поскольку современные системы диалога защищены от ошибок - уточняют, просят или переводят клиентов на живых операторов, если они не понимают. Впрочем, важно измерять число ошибок, следить за ними и научить бота осознавать запросы, с которыми он столкнулся. В противном случае популярность бота будет падать, а мы не будете знать, что реальная причина кроется в простом непонимании.

GCR - Goal Completion Rate. Это ещё одна метрика, уникальная для всех систем диалога. Любой разговор с ботом направлен на достижение определенной цели. Это могут быть цели, поставленные самим пользователем, например, для пополнения баланса или получения ответа на вопрос.

Система управления ботом должна иметь возможность следить за достижением целей. Анализируя эти данные, мы можем найти в диалоге моменты, когда разговор прекращается, хоть и цель не достигнута. На эти моменты следует обратить внимание, а также переписать скрипт.

Все вышеуказанные метрики измеряемые. Впрочем, преимуществом чат - ботов является то, что они используют не просто ссылки и кнопки, а именно слова. Таким образом, кроме метрик рекомендуется использовать систему семантического анализа реплик пользователей. Это поможет понять эмоции разговора, узнать о том, какие запросы наиболее популярны у пользователей, а также получить большое количество информации о ЦА, которые потом могут использоваться для маркетинговых кампаний.

Вывод

Чат-боты оптимизируют рутинные процессы и, по большому счету, повышают конверсию. Для внедрения чат-бота в воронку нужно следовать следующим принципам: простота использования, решение одной из главных проблем, ненавязчивость и польза для конечных клиентов. Придерживаемся этих правил для оптимизации расходов и увеличения прибыли.

3. **Заключение**   
Основные результаты проекта

Основные результаты проекта по разработке Телеграм бота для парсинга сайтов:

* Создан Телеграм бот, который может парсить данные с различных сайтов и предоставлять их пользователю в удобном формате.
* Бот был протестирован и показал высокую эффективность в работе.
* Бот был интегрирован с другими сервисами и приложениями для расширения его функциональности.  
  В целом, проект по разработке телеграм бота для парсинга сайтов является успешным и позволяет автоматизировать процесс сбора и анализа информации.

Выводы и рекомендации по использованию телеграмм бота в практике

Выводы:

– Телеграмм бот для парсинга сайтов может быть эффективным инструментом для сбора и анализа данных с различных сайтов.  
– Использование ботов позволяет автоматизировать процессы сбора и обработки информации, что упрощает работу и повышает ее эффективность.  
– Для достижения наилучших результатов необходимо использовать методы парсинга и анализа, которые подходят для конкретных задач.  
– Необходимо проводить тестирование бота перед его использованием в практических целях.

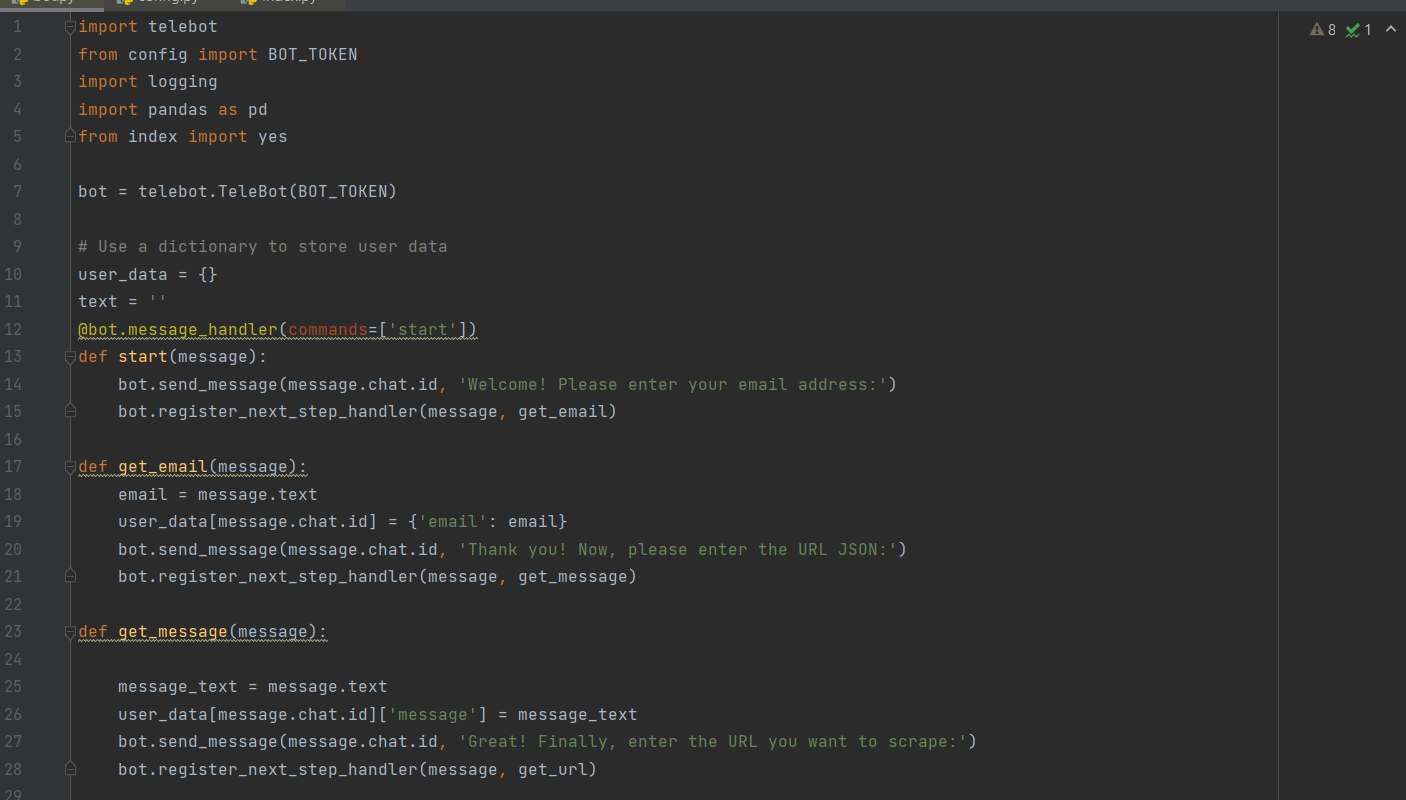
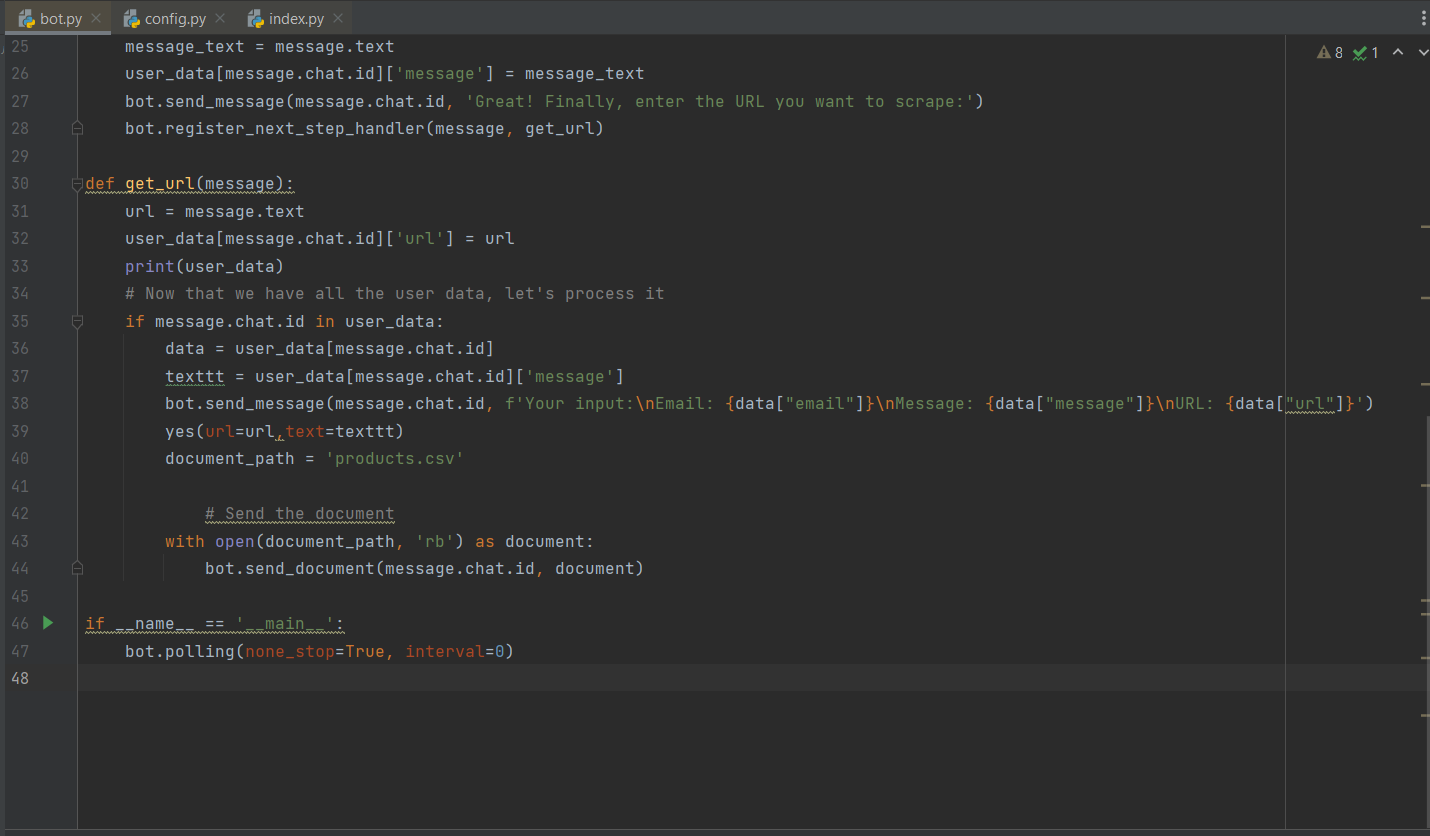
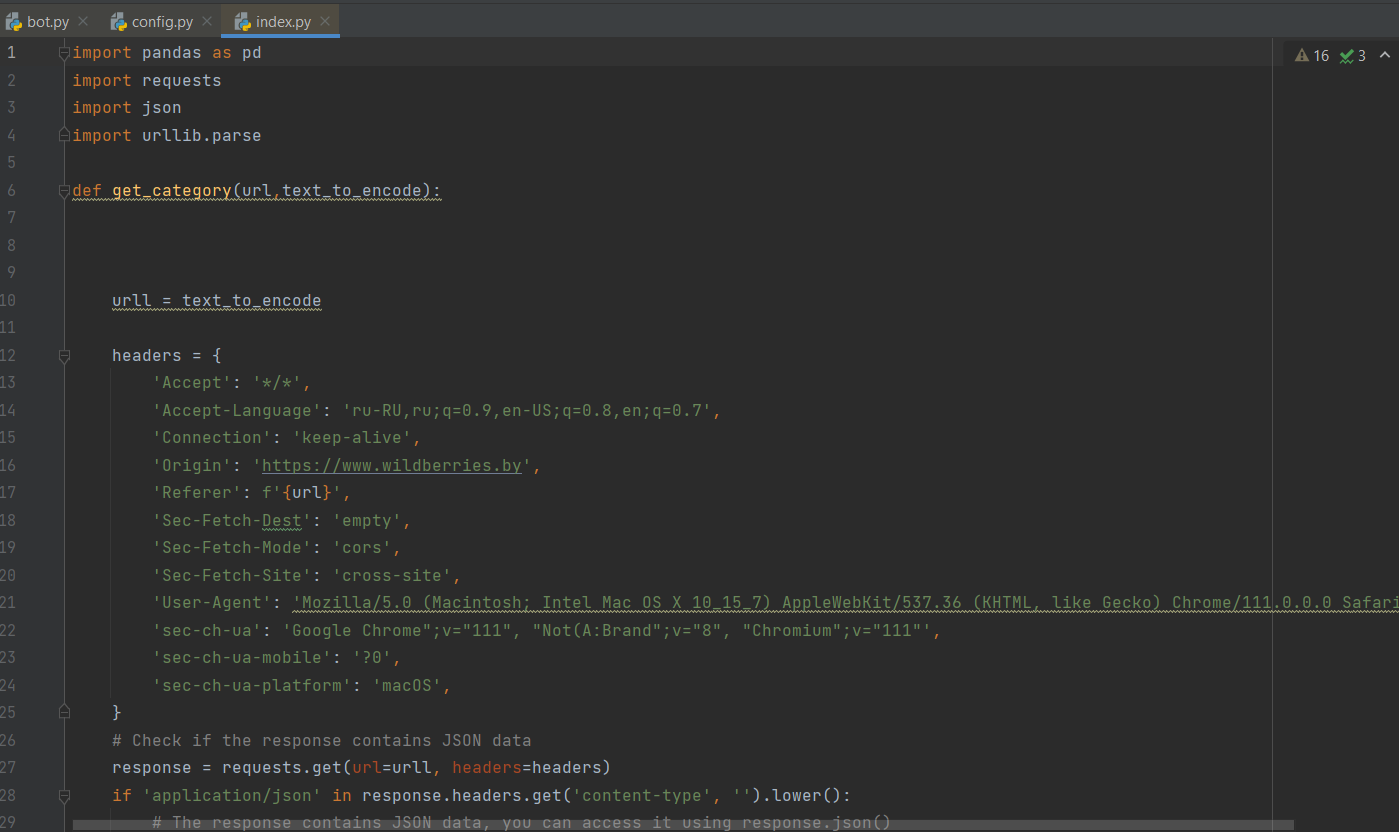
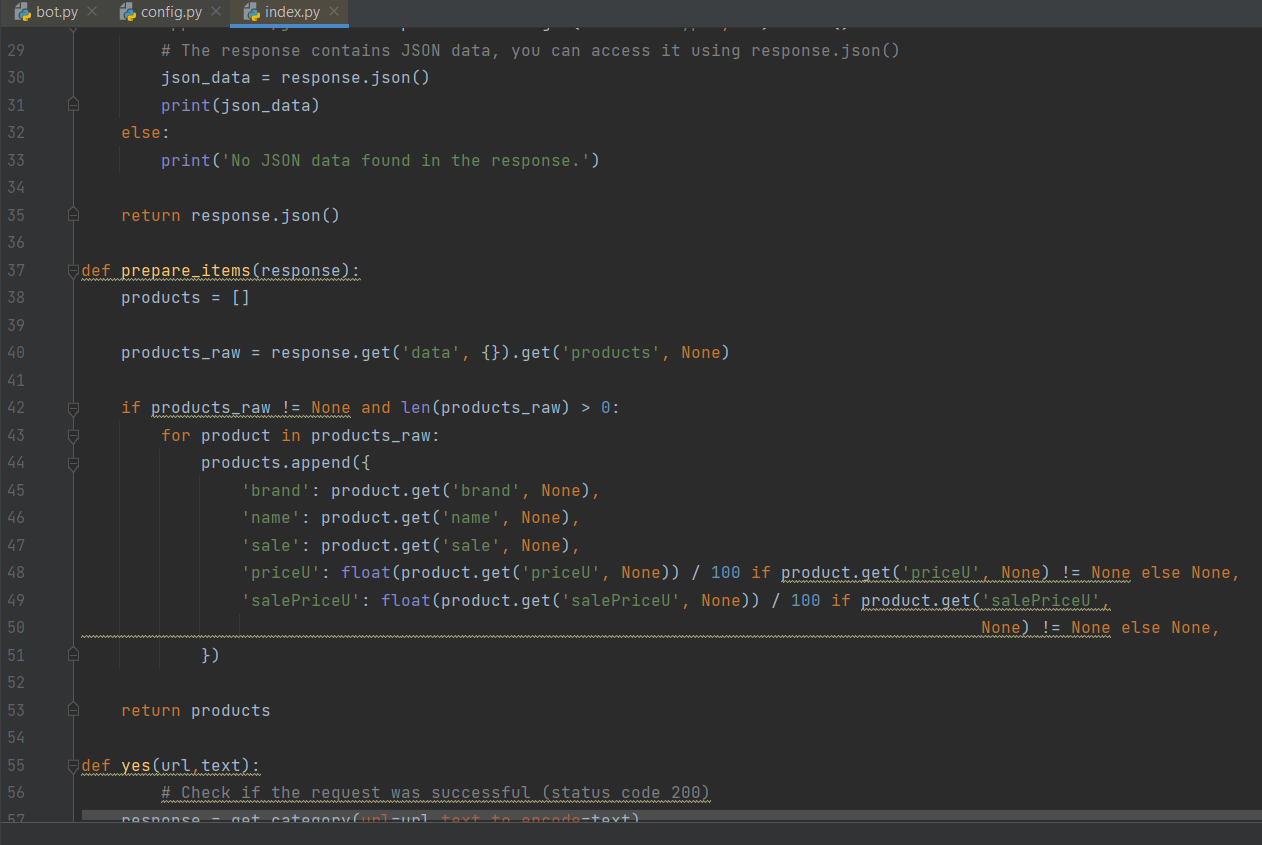
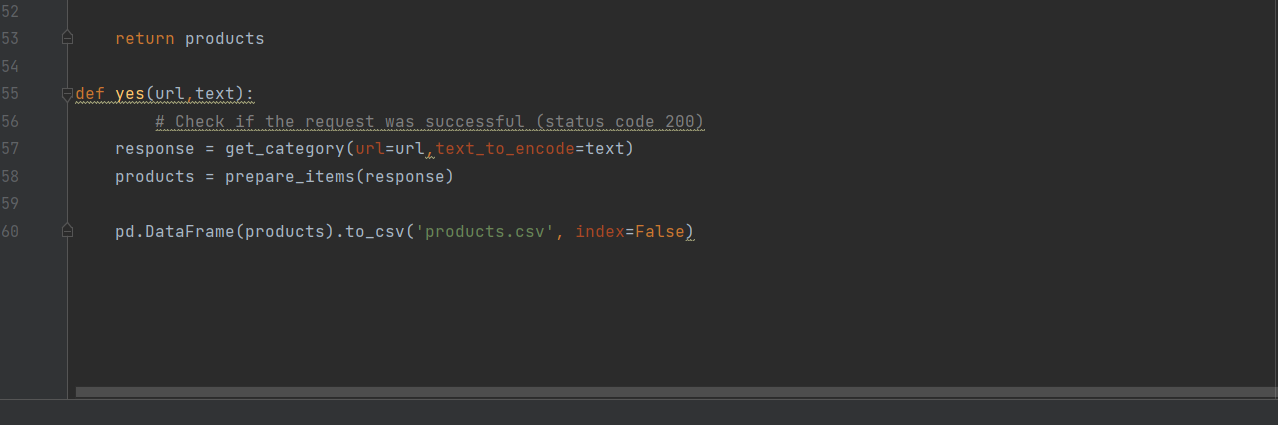
Рекомендации:

– При разработке телеграмм бота для парсинга сайтов следует учитывать требования безопасности и защиты данных пользователей.  
– Важно обеспечить удобный интерфейс для пользователей, чтобы они могли легко использовать бот и получать нужную информацию.  
– Рекомендуется использовать методы машинного обучения для улучшения качества анализа данных и повышения точности результатов.  
– Стоит обратить внимание на скорость работы бота и оптимизировать его, если это необходимо.  
– Нужно регулярно обновлять и улучшать бота для соответствия современным требованиям и вызовам рынка.

Перспективы развития

Перспективы развития телеграмм бота для парсинга сайтов могут быть следующими:

Расширение функциональности бота: добавление новых функций, таких как анализ социальных сетей, работа с изображениями и видео, интеграция с другими сервисами.  
Улучшение методов парсинга: использование более современных инструментов и алгоритмов, таких как Deep Learning, для повышения точности и скорости работы бота.  
Оптимизация работы бота: улучшение производительности, снижение времени отклика, повышение надежности и безопасности.  
Интеграция с другими платформами и сервисами: расширение возможностей бота за счет интеграции с социальными сетями, аналитическими инструментами, CRM системами и другими сервисами.

**Список использованных источников.**[Библиотека Python разработчика – Telegram](https://t.me/s/BookPython/1872)  
[Разработка чат-ботов и разговорных интерфейсов | Джанарсанам Срини, Райтман Михаил Анатольевич | Электронная книга — купить в интернет-магазине OZON с быстрой доставкой](https://www.ozon.ru/product/razrabotka-chat-botov-i-razgovornyh-interfeysov-dzhanarsanam-srini-raytman-mihail-anatolevich-934531754/?asb=8mItCkpTNuBfvI%252BlfXcptNRn6eVLDU3n%252B2r1qw%252BR%252FPk%253D&asb2=somIl_y-v8jhhNX0PsPDNxm9wMxrDSNrbqJeO8qU3A9epmZ49KMM501z74Uan15M&avtc=1&avte=2&avts=1692429425&keywords=%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0+%D1%82%D0%B5%D0%BB%D0%B5%D0%B3%D1%80%D0%B0%D0%BC+%D0%B1%D0%BE%D1%82%D0%BE%D0%B2+%D0%BA%D0%BD%D0%B8%D0%B3%D0%B8&sh=xzAAlUZ0uQ)  
[Бот впомощь. Книга-практикум по созданию чат-ботов и автоматизации бизнеса | Бушмелева Ксения Григорьевна | Электронная книга — купить в интернет-магазине OZON с быстрой доставкой](https://www.ozon.ru/product/bot-vpomoshch-kniga-praktikum-po-sozdaniyu-chat-botov-i-avtomatizatsii-biznesa-bushmeleva-913134845/?asb=qWDSzlIL%252Fr89Dv4RwX9pjEGobH4xDEOa0xlrsOmjrOg%253D&asb2=aqFKj4CBqRerVGiK9HwTvbJ6DKhAqmKtGT5qe4xU-CKau0guG6x4HeDYrSZ9JAKu&avtc=1&avte=2&avts=1692429478&keywords=%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0+%D1%82%D0%B5%D0%BB%D0%B5%D0%B3%D1%80%D0%B0%D0%BC+%D0%B1%D0%BE%D1%82%D0%BE%D0%B2+%D0%BA%D0%BD%D0%B8%D0%B3%D0%B8&sh=xzAAlUeGDQ)  
[Читать книгу «Telegram Bot. Руководство по созданию бота в мессенджере Телеграм.» онлайн полностью📖 — Артема Демиденко — MyBook.](https://mybook.ru/author/artem-demidenko/telegram-bot-rukovodstvo-po-sozdaniyu-bota-v-messe/read/?ysclid=llhotru0hs140265198)[Введение - Пишем Telegram-ботов с aiogram 3.x (β) (mastergroosha.github.io)](https://mastergroosha.github.io/aiogram-3-guide/)[https://reyting-marketpleysov.ru/](http://textovod.com/unique/link?url=https%3A%2F%2Freyting-marketpleysov.ru%2F&key=5eaebcf497b54c823cbe2a66c1150aeb)  
[Парсинг - синтаксический анализ. Экопарк Z (ep-z.ru)](https://ep-z.ru/stroitelstvo/sayt/python/parsing)  
[Для чего нужен питон (gshimki.ru)](https://gshimki.ru/tehnologija/27/dlja-chego-nuzhen-piton)  
[Что такое Python? – Описание языка программирования Python – AWS (amazon.com)](https://aws.amazon.com/ru/what-is/python/)  
[PyCharm: что это за среда разработки, как её установить и использовать / Skillbox Media](https://skillbox.ru/media/code/pycharm-kak-eye-ustanovit-i-ispolzovat/)  
[Руководство telegram bot api (instrukciyarus.ru)](https://instrukciyarus.ru/rukovodstvo-telegram-bot-api/)  
[Как вытащить данные с сайта python (businessclass.com)](https://www.businessclass.com.ua/informacija/49/kak-vytashhit-dannye-s-sajta-python/)  
 **Приложения.  
 Код скрипта на Python.** bot.py  
  
  
 index.py  
  
  
  
  
  
  
  
  
  
  
 **Пошаговая инструкция работы бота**  
  
