Лабораторная работа №1. Разработка чат-бота.

Цель: разработать чат-бот на языке Python, выполняющий заданные действия по варианту задания.

Задачи:

1. разработка регулярного выражения для поиска в произвольном тексте команд чат-боту и параметров выполнения команд;
2. реализация требуемых действий чат-бота в предварительной заготовке программы на языке Python;
3. разработка заготовки самого чат-бота, принимающего сообщения от пользователя и отправляющего сообщения пользователю;
4. компоновка всех перечисленных выше этапов в рамках чат-бота.

Протокол к лабораторной работе должен содержать описание всех задач, с приведением полученного регулярного выражения, алгоритма в виде блок-схемы реализации логики работы чат-бота, с расшифровкой поддерживаемых команд и с пояснениями к выполняемыми действиями, никнейма чат-бота в мессенджере для тестирования на отчёте и скриншотов диалога с чат-ботом с демонстрацией выполнения заданных действий чат-ботом. Также в архив необходимо приложить .ру-файл исходного кода чат-бота.

ВСЕ ВЫПОЛНЕННЫЕ РАБОТЫ ДОЛЖНЫ БЫТЬ ЗАГРУЖЕНЫ НА EOS2,

Общие требования к чат-боту:

- чат-бот должен выполнять требуемые в задании действия по вводу команд, задаваемых пользователем, с опциональными параметрами для указания особенностей выполнения заданных действий;
- некоторые параметры команды чат-боту могут быть опциональными, т.е. если их не указать, подразумевается значение по умолчанию, заданное студентом;
- чат-бот может принимать одну или несколько различных команд для реализации выполнения задания курсовой работы;
- чат-бот не должен реагировать на сообщения-шум, которые не являются зарезервированными командами чат-бота;
- поиск в тексте сообщения команд и параметров необходимо реализовать с помощью регулярных выражений;

- если для команды не хватает каких-то параметров или они указаны неверно, чат-бот может сообщать об ошибке или не выполнять действия, но он должен говорить обо всех ошибках, возникающих в процессе выполнения заданных действий по команде, если они возникли по какой-то причине;
- чат-бот должен выдавать ответ на заданную команду об успешности или неуспешности выполнения команды, результаты выполнения команды или выводить требуемую пользователем информацию по команде (в зависимости от задания);
- внешний вид и содержание сообщений с информацией, выдаваемой чат-ботом задаётся студентом самостоятельно, однако они должны быть понятны, читаемы, опрятно оформлены;
- чат-бот должен проверять корректность формата вводимых данных (ФИО, почтовый адрес, адрес электронной почты, номер телефона и т.д.), логичности данных (например, не указывать дату позднее или ранее, чем текущий момент, по заданию, т.е. не вносить данные из будущего, не делать уведомления в прошлое и т.п.) и выводить сообщение об некорректности вводимых данных или игнорировать команду (упрощенный вариант);
- для хранения данных в файле можно использовать формат JSON.

Варианты заданий

1. Реализовать отправку письма на электронную почту с помощью чат-бота. В параметрах команды указать тему письма, адреса получателя (один и более), текст письма и время отправки в течение дня или «сейчас» (порядок на усмотрение студента). О работе с электронной почтой в Python можно посмотреть, например, здесь: https://python-scripts.com/send-email-smtp-python, https://realpython.com/python-send-email/ или в справке: https://docs.python.org/3/library/email.examples.html. Для реализации отправки писем необходимо создать свой аккаунт в почтовом сервисе. Пример возможного варианта команды: «/send Привет! Как дела? Как погода? -t Тестовое письмо -r ivanov@example.com, petrov@example.com -w сейчас»

2. Реализовать отправку чат-ботом напоминаний, создаваемых пользователем. В параметрах команды указать текст, о чём надо напомнить, когда (разные варианты параметра: через заданный период времени или в указанное время, дату, или «сегодня, завтра»), делать ли напоминание ежедневно. Напоминание может быть не менее, чем через 10 секунд. Здесь понадобятся модули для работы со временем (time, datetime). Реализовать функционал, позволяющий пользователю запросить список всех созданных напоминаний и удалять те, которые не нужны (например, по номеру в списке). Пример возможного варианта команды добавления напоминания: «/add Купить хлеба -t завтра 10:00 no repeat»

3. Реализовать с помощью чат-бота возможность отслеживания своих доходов и расходов. Пользователь отправляет чат-боту сообщение с указанием, на что он потратил деньги, когда (дата или «сегодня» и время), но не позднее текущей даты, какую сумму (в рублях), а также откуда он получил деньги (также с отметкой, когда и какую сумму). Данные сохраняются в файл. Пользователь может запросить вывод списка всех расходов и доходов с указанием баланса на текущий момент по имеющемуся списку и очистить файл. Пример возможного варианта команды добавление расходов: «/spend Хлеб, молоко -c 85.50 -d 01.01.2020»

4. Реализовать генерацию результатов бросков игровых кубиков чат-ботом (в частности для игр в DnD, по аналогии с ресурсом https://rolz.org/). Пользователь вводит команду в формате: <кол-во_кубов1>d<кол-во_граней1> +/- … +/- <кол-во_кубовN>d<кол-во_гранейN> +/- <любое число> (в произвольном порядке), а чат-бот выводит полученное значение суммы/разности случайно сгенерированных чисел. Если количество кубов не указано перед «d», то считается, что куб один. Пример возможного варианта команды: «2d6 + d10 – 10 + 3d20»

5. Реализовать записную книжку в чат-боте. Пользователь может заносить в чат-бот информацию о людях: номер мобильного телефона, ФИО, перечисление учетных записей в мессенджерах/соц. сетях (ни одной или одна и более, через запятую). Данные сохраняются в файл. Пользователь может запросить вывод списка всех записей людей, отсортированных по ФИО, удалить выбранную

запись, задавая номер в списке, и очистить файл. Пример возможного варианта команды: «/add 89992233223 Иванов Иван, vk:ivanivan, t:ivan_v_telege»

6. Реализовать получение прогноза погоды от чат-бота. Прогноз может выдаваться на указанную дату (или «сегодня, завтра»), на период времени с одной даты по другую, пользователь может указать город для прогноза, прогноз может быть коротким или подробным (содержание каких данных в прогнозе – на усмотрение студента). Пример возможного варианта команды: «/weather –t завтра –l г. Волгоград -full». Для реализации получения прогноза погоды понадобится работа с API какого-либо сервиса, например: https://rapidapi.com/blog/weather-api-python/

7. Реализовать сохранение закладок по тематикам с помощью чат-бота. Пользователь может добавлять тематику. Потом для каждой из добавленных тематик он может сохранять адреса различных сайтов, статей, которые хочет прочесть позже. Данные сохраняются в файл. После он может запросить, что прочесть по заданной тематике и чат-бот выводит список ссылок. Пользователь может удалять ссылку из тематики, а также тематику целиком (со всеми ссылками в ней). Пример возможного варианта команды добавления закладки: «/add Программирование на Python -u https://example.com». Если такой тематики нет, она создаётся.

8. Реализация чат-бота собеседника с помощью библиотеки nltk, пакета chat. Необходимо заготовить словарь ответов по заданным регулярным выражениям и, используя пакет chat библиотекb nltk выдавать ответы пользователю. Подробнее можно прочесть здесь: https://medium.com/datadriveninvestor/build-your-own-chat-bot-using-python-95fdaaed620f, https://towardsdatascience.com/build-your-first-chatbot-using-python-nltk-5d07b027e727 пакет chat: https://www.nltk.org/api/nltk.chat.html. Необходимо использовать метод chat.response(<строка сообщения от пользователя>). Пример возможного варианта команды: «-Привет! Как дела? –Нормально! А у тебя? –Тоже хорошо! –Это здорово!»

9. Реализовать получение рекомендаций от чат-бота о том, что приготовить из заданного набора продуктов, имеющихся в

холодильнике у пользователя. Данные хранятся в файле в формате JSON, он заполняется студентом заранее. Формат представлен ниже:

```
{
        “dish”: “<наименование блюда>”,
        “type”: “<завтрак/обед/ужин/перекус>”,
        “ingredients”: [
                “<ингредиент 1>”,
                …
                “<ингредиент N>”
        ]
}
```

Пользователь вводит список продуктов, которые у него имеются в наличии, тип блюда и чат-бот возвращает список доступных блюд. Пример возможного варианта команды: «/dish -t перекус -i хлеб, масло, колбаса, сыр, молоко, яйца»

10. Реализовать напоминания о днях рождения с помощью чат-бота. Пользователь добавляет в список дней рождений записи, в которых содержатся ФИО, дата рождения и почтовый адрес места обычного места встречи/проживания именинника. Чат бот должен по утрам напоминать, что у кого-то из списка через день состоится день рождения, если есть такая запись. Пользователь может запросить список записей о днях рождения и удалить одну из записей по номеру в списке. Пример возможного варианта команды добавления записи о дне рождения: «/add –p Иванов Иван –w 01.01.1990 –l ул. Пушкина, 22, кв. 34»
11. Реализовать чат-бот для проведения голосования. Пользователь создаёт голосование, задавая вопрос для голосования, варианты выбора, дату и время окончания голосования. Идентификаторы пользователей, которые инициировали беседу с чат-ботом, должны храниться в файле, не повторяясь. Чат-бот рассылает всем пользователям из файла и текущему пользователю опрос: вопрос голосования и пронумерованные варианты выбора. Пользователи голосуют, отправляя номер варианта. По окончании голосования в заданные дату и время чат-бот выводит процентные соотношения количеств голосов за каждый из вариантов. Каждый пользователь, кто добавил себе чат-бота может создать только одно текущее

голосование. Он может его остановить по отдельной команде с выводом набранных результатов с пометкой, что оно остановлено. Другие пользователи могут проголосовать только один раз. Пример возможного варианта команды создания голосования: «/poll Где сегодня встречаемся нашей компанией друзей? -a В парке -a В кафе -a На набережной -a У Ивана дома -till 01.01.2020 15:00»

12. Реализовать чат-бот для записи на консультацию по предмету в университете. Студент запрашивает список доступных консультаций на неделю, чат-бот ему его выводит. Список выводится в формате: название предмета, дата, время, аудитория. Затем студент пишет чат-боту команду на запись на консультацию, в которой содержится: его ФИО, группа, предмет, дата и время. Чат-бот выдаёт ответ об успешной записи на консультацию, если в имеющемся списке консультаций на неделю есть выбранная консультация с таким предметом, датой и временем. Данные о консультациях хранятся в файле в формате JSON, он заполняется вами заранее. Формат представлен ниже:

```
[
    {
        “class”: “Наименование предмета консультации 1”,
        “date”: “Дата и время консультации 1”,
        “room”: “аудитория и корпус 1 (например, В-1001)”
    },
    …
    {
        “class”: “Наименование предмета консультации N”,
        “date”: “Дата и время консультации N”,
        “room”: “аудитория и корпус N (например, В-1001)”
    }
]
```

Пример возможного варианта команды записи на консультацию: «/addme Иванов Иван Иванович -r ВТВ-268 -c Мат. анализ -d 01.01.2020 15:00»

13. Реализовать чат-бот для записи своих результатов выполнения упражнений в тренажерном зале. Пользователь отправляет чат-боту сообщение с указанием даты (не ранее текущей даты), какое упражнение он делал, какой вес брал, сколько подходов по сколько повторений. Вес может быть опциональным, чтобы учитывать тренировки без веса. Данные сохраняются в файл. Пользователь может запросить вывод списка всех записей, сгруппированных по датам и очистку файла. Пример возможного варианта команды добавление расходов: «/add 01.01.2020, Жим лёжа, 4, 12, 60» (вес добавляется в конце)

Особенности чат-ботов в Telegram

Мессенджер Telegram позволяет его пользователям создавать своих чат-ботов. В нём, и в других мессенджерах Бот — это не просто «автоответчик», его правильнее считать автоматизированным помощником. Это многофункциональный инструмент, как для бизнеса, так и для развлечений. Боты могут использоваться для:

- индивидуальных уведомлений и контента
- интеграции с другими сервисами и получения от них оповещений (почта, переводчик, погода, новости, YouTube и др.)
- создания пользовательских инструментов, игр и др.

В данной работе рассматриваются чат-боты для Telegram, поскольку их создание является одним из самых простых и позволит сфокусироваться на реализации логики работы бота, а не на его создании и подготовке к работе.

Telegram Bots – это специальные учетные записи, для настройки которых не требуется дополнительный номер телефона. Пользователи могут взаимодействовать с ботами двумя способами:

- отправлять сообщения и команды ботам, открывая с ними чат или добавляя их в группы
- отправлять запросы из любого чата, набрав @username бота и запрос. Это позволяет отправлять контент из встроенных ботов прямо в любой чат, группу или канал

Сообщения, команды и запросы, отправленные пользователями, передаются программному обеспечению, написанному вами для бота, работающему на вашем компьютере (сервере). Сервер-посредник Telegram обрабатывает все задачи шифрования и связи с Telegram API. Вы общаетесь с этим сервером через простой HTTPS-интерфейс, который предлагает упрощенную версию Telegram API – Bot API. Подробное его описание здесь: https://core.telegram.org/bots/api

Особенности ботов в Telegram:

- у ботов нет онлайн-статуса и последних посещенных меток времени, вместо этого на интерфейсе отображается метка «бот»
- у ботов ограниченное облачное хранилище - старые сообщения могут быть удалены сервером вскоре после их обработки
- боты не могут инициировать разговоры с пользователями. Пользователь должен либо добавить их в группу, либо сначала

отправить им сообщение. Люди могут использовать ссылки t.me/<bot_username> или поиск по имени пользователя, чтобы найти вашего бота

- имена пользователей ботов всегда должны заканчиваться на «бот» (например, @TriviaBot, @GitHub_bot)
- при добавлении в групповой чат по умолчанию боты не получают все сообщения от пользователей к пользователям (только в особенных случаях, см. в справке), а получают только:
    - сообщения, начинающиеся с косой черты ‘/’ (см. ниже и в справке)
    - ответы на собственные сообщения бота
    - служебные сообщения (люди добавлены или удалены из группы и т. д.)
    - сообщения с каналов, на которых они зарегистрированы
- если в группе несколько ботов, можно добавить имена команд ботов в команды, чтобы избежать путаницы:
    - /start@TriviaBot
    - /start@ApocalypseBot

Чтобы облегчить пользователям использование ботов, разработчики Telegram просят всех разработчиков поддерживать у своих ботов несколько основных команд. Приложения Telegram будут иметь ярлыки интерфейса для этих команд:

- /start - начало взаимодействия с пользователем, например, отправка приветственного сообщения
- /help - возвращает справочное сообщение. Это может быть краткий текст о том, что может делать ваш бот, и список команд
- /settings - (если применимо) возвращает настройки бота для этого пользователя и предлагает команды для редактирования этих настроек

Создание чат-бота в Telegram

Сперва нужно зарегистрироваться в Telegram. Наиболее удобно использовать веб-клиент для знакомства с основными принципами работы ботов и API: https://web.telegram.org/

Для создания ботов есть специальный бот BotFather (https://t.me/botfather), откройте приложение, найдите @BotFather, начните

с ним беседу и выполните несколько простых шагов, инструкция здесь: https://core.telegram.org/bots#6-botfather. После того, как вы создали бота и получили свой **токен авторизации (можно сказать, уникальный идентификатор бота)**, перейдите к руководству по Bot API, чтобы узнать, чему вы можете научить своего бота. **Токен нужно держать в секрете, чтобы никто не мог получить доступ к вашему боту и не начал его использовать в своих корыстных целях.** В случае чего, его можно пересоздать или отозвать (см. документацию у BotFather).

Примеры ботов (в том числе и на Python) приведены здесь: https://core.telegram.org/bots/samples

Вам нужно начать беседу с вашим ботом. Найдите своего бота в поиске пользователей. Введите в поисковой строке его имя и нажмите на кнопку /start. Пока что ваш бот ничего не умеет, но уже может принимать сообщения. Отправьте сообщение, например, «Привет». Это первое сообщение очень важно, поскольку оно станет первым обновлением, которое получит ваш бот.

Все запросы к Telegram Bot API должны обслуживаться по HTTPS и должны быть представлены в этой форме:

https://api.telegram.org/bot<token>/ИМЯ_МЕТОДА_ЗАПРОСА

Если вы в первый раз работаете с API, то разобраться вам поможет браузер. Откройте новую вкладку и воспользуйтесь Telegram API, отправив в строку браузера запрос:

https://api.telegram.org/bot<ваш_токен>/getUpdates

Таким образом, вы отправляете Get-запрос на сервер Telegram для вашего бота.

По сути, каждый раз, когда вам нужно получить, обновить или удалить информацию, хранящуюся на сервере, вы отправляете запрос и получаете ответ.

Подробнее здесь: https://ru.wikipedia.org/wiki/HTTP

Про запрос методом Get здесь: https://webkyrs.info/post/http-zapros-metodom-get

Отправлять данные боту обратно будем через Post-запрос: https://ru.wikipedia.org/wiki/POST_(HTTP)

Открыв этот адрес в браузере, вы отправите запрос на сервер Telegram, и он ответит вам в формате JSON, который разбирался на прошлой лабораторной работе.

Если вы своему боту ничего не писали, то там будет такая строка:

```
{"ok":true,"result":[]}
```

Т.е. бот работает, но он ещё ничего не получал, поэтому массив “result” пуст. Если «ok» равно true, запрос был успешным, и результат запроса можно найти в поле «result». В случае неудачного запроса «ok» равно false, а ошибка объясняется в «description».

В браузере JSON-строка от бота отображается сплошной строкой. Можете поместить этот текст в сервис JSON-форматтера, как говорилось на предыдущей лабораторной работе, чтобы посмотреть, как она выглядит с расстановкой переносов и подсветкой синтаксиса. Вы увидите что-то вроде такого:

```
{
 "ok":true,
 "result":[{
 "update_id":523349956,
 "message":{
 "message_id":51,
 "from":{
        "id":303262877,
        "first_name":"YourName"
 },
 "chat":{
        "id":303262877,
        "first_name":"YourName",
        "type":"private"
 },
```

```
"date":1486829360,
"text":"Hello"
}
}]
}
```


Назначение содержимого этого JSON-объекта разобрано в документации: https://core.telegram.org/bots/api#making-requests

Словарь обновлений, возвращаемый ботом, состоит из двух элементов: ok и results. Нас интересует вторая часть – список всех обновлений, полученных ботом за последние 24 часа.

Подробнее о методе getUpdates: https://core.telegram.org/bots/api#getupdates

Чтобы отправить сообщение от лица бота, нужно отправить запрос /sendMessage (см. в документации: https://core.telegram.org/bots/api#sendmessage). Вы увидите, что он принимает два параметра: chat_id и text. Т.е. мы указываем кому бот отправляет сообщение и с каким текстом. Вы можете создавать цепочки параметров в адресной строке браузера, используя ? для первого и & для всех последующих параметров. Команда для отправки сообщения будет выглядеть так:

https://api.telegram.org/bot<token>/sendMessage?chat_id=303262877&text=Hello

Попробуйте получить ответ от вашего бота, подставив в качестве chat_id значение вашего идентификатора чата, полученное после вызова /getUpdates (в нашем примере — 303262877). Текст сообщения может быть любым.

Программирование логики бота

Если у вас имеется возможность использовать мессенджер Telegram на компьютере, то вы можете разрабатывать его логику с помощью Python и среды разработки, установленных у вас на ПК. В противном случае вы будете получать ошибку о недоступности ресурса Telegram по понятным причинам (пример решения можно посмотреть в этой статье:

https://habr.com/ru/post/448310/). В этом случае вам необходимо использовать онлайн-блокнот, о которых говорилось на второй лабораторной работе. И в этом случае лучше использовать блокнот, доступный только вам, например, в Google Colab (https://colab.research.google.com/), опять же, для сохранности вашего токена.

В случае работы на вашем компьютере убедитесь, что у вас установлен pip, обычно он устанавливается вместе с Python (https://pypi.org/project/pip/)

Необходимо установить пакет requests при помощи pip, вызываемого в консоли Windows (или см. документацию к онлайн-блокноту, а, например, в Google Colab он уже установлен, можно пропустить этот шаг):

$ pip install requests

Мы будем использовать данный модуль, больше о нём можно прочесть здесь: https://requests.readthedocs.io/en/master/

Напишем скрипт, который будет проверять обновления и отвечать на новые сообщения.

Сперва бот должен проверить обновления. Первое сообщение можно расценивать как самое свежее, но getUpdates возвращает все обновления за последние 24 часа. Напишем небольшой скрипт, чтобы получить самое последнее обновление, доставать chat_id из обновления и отправлять сообщение.

```
import requests


# Адрес нашего бота (без указания запроса)

url = 'https://api.telegram.org/bot<токен_вашего_бота>/'


# Получаем JSON-строку ответа на запрос getUpdates

def get_updates_json():

        response = requests.get(url + 'getUpdates')     # Выполняем
Get-запрос, возвращается объект ответа
```

```
    return response.json()                                  # Из объекта ответа берём JSON


# Получить последнее событие от бота

def last_update(response_json):

    results = response_json['result']          # Обращаемся к словарю по ключу result

     last_update_index = len(results) - 1      # Последнее событие - последнее по порядку минус 1, т.к. нумерация с нуля

     return results[last_update_index]          # Возвращаем последнее событие


# Получаем идентификатор чата, от которого пришло сообщение

def get_chat_id(result_json):

    return result_json['message']['chat']['id']      # Смотрим по JSON строке: message -> chat -> id


# Отправляем пришедшее событие назад

def send_message(chat_id, text):

    params = {'chat_id': chat_id, 'text': text}              # Формируем параметры запроса: куда и что отправяем

        return requests.post(url + 'sendMessage', data=params)    # Отправляем Post-запрос, возвращаем объект ответа


# Каскадно выполняем запрос к боту, получаем последнее событие и id чата отправителя

current_chat_id = get_chat_id(last_update(get_updates_json()))

send_message(current_chat_id, "Привет!")
```

Помните, как мы объединяли параметры при помощи «?» и «&»? Вы можете сделать то же самое, добавив словарь в качестве второго дополнительного параметра в функциях get/post из пакета requests.

Скрипт готов, но он не идеален. Главным минусом является необходимость запускать его каждый раз, когда мы хотим, чтобы бот отправил сообщение. Исправим это. Чтобы бот слушал сервер и получал обновления, нам нужно запустить основной цикл. На новой строке, после import requests, добавьте

from time import sleep

Из модуля работы со временем мы импортируем функцию sleep – пауза работы скрипта на указанное количество секунд.

После этого замените две последние строки на следующий код:

```
# Функция - точка входа в программу

def main():

    last_update_id = get_update_id(last_update(get_updates_json())) # Получаем идентификатор самого последнего события



    # Бот работает в вечном цикле, нужно будет принудительно останавливать свою программу

  while True:

      updates_json = last_update(get_updates_json())       # Получаем последнее событие на текущий момент

     current_update_id = get_update_id(updates_json)    # Получаем его идентификатор



     # Если было новое событие, отправляем сообщение и запоминаем его идентификатор

    if current_update_id != last_update_id:

      send_message(get_chat_id(updates_json), "Привет!")

      last_update_id = current_update_id
```

```
    sleep(1)    # Ждём одну секунду, не забудьте про эту функцию, чтобы наш код не слал постоянно большое количество запросов на сервер Telegram


if __name__ == '__main__':

    main()
```

Хотя мы и добавили таймаут в 1 секунду, пример выше можно использовать только в обучающих целях, поскольку он использует частые опросы (short polling). Это плохо влияет на сервера Telegram, поэтому их нужно избегать. Если мы будем использовать способ получения обновлений через getUpdates без параметров, то запросы будут происходить слишком часто.

Есть ещё два способа получения обновлений через API — длинные опросы (long polling) и вебхуки (webhooks). Общая разница между polling и webhooks:

- **Polling** (через get_updates) периодически подключается к серверам Telegram для проверки новых обновлений. Мы отправляем запрос, получаем ответ.
- **Webhook** - это URL, который вы передаете Telegram один раз. Когда приходит новое обновление для вашего бота, Telegram отправляет это обновление по указанному URL.

Для реализации работы через webhooks есть свои определённые задачи, которые нужно выполнить, подробнее можно почитать здесь: https://github.com/python-telegram-bot/python-telegram-bot/wiki/Webhooks, но мы пока будем пользоваться polling.

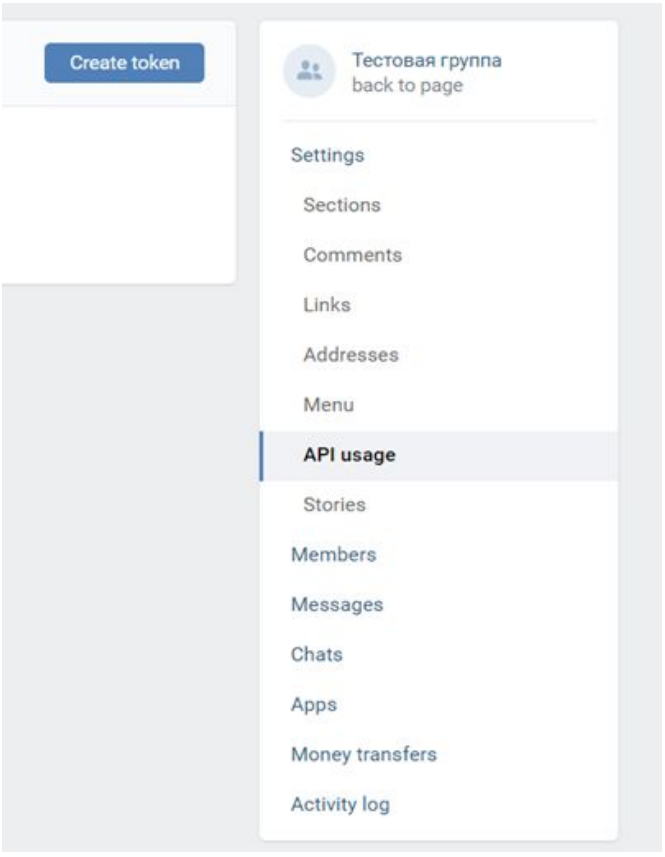
Long polling отличается от short polling тем, что в параметре timeout метода getUpdate есть ненулевое значение (https://core.telegram.org/bots/api#getupdates). Т.е. когда мы делаем данный запрос с параметром timeout=60, результат будет возвращён сразу, если есть новые обновления, или он будет ожидать 60 секунд пока обновления не появятся и только после этого вернёт значение (либо обновление появились, либо так и не появились).

Поскольку мы начали использовать в скрипте основной цикл, мы должны переключиться на длинные опросы. Сперва изменим первую функцию, добавив в неё параметр timeout. Сам по себе он не уменьшит частоту проверки обновлений и будет работать только в том случае, когда обновлений нет. Чтобы помечать уже просмотренные обновления, нужно добавить параметр сдвига offset (тоже пока использовать не будем):

```
# Получаем JSON-строку ответа на запрос getUpdates

def get_updates_json():

     params = {'timeout': 100, 'offset': None}       # Ждём ответа на протяжении заданного в параметре timeout числа секунд

       response = requests.get(url + 'getUpdates', data=params)    # Выполняем Get-запрос, возвращается объект ответа

    return response.json()                              # Из объекта ответа берём JSON
```

Чат-бот для ВКонтакте

Если у кого-то возникли затруднения с созданием чат-бота для Telegram или работой с ним из-за блокировок, альтернативой является создание чат-бота для соц. сети ВКонтакте. Наиболее простым вариантом будет создать чат-бота сообщества (группы). Для этого, соответственно, необходимо создать для себя тестовое, можно закрытое, сообщество. В его настройках необходимо сделать доступными сообщения сообщества. Процесс создания данного чат-бота будет похожим на тот, что расписан для чат-бота в Telegram. Access token, аналогично, необходимо получить в настройках группы в разделе «Использование API» (описано в статьях по ссылкам ниже).



В остальном понадобится также только Python.

Работа с VK API достаточно подробно расписана в справке:

- https://vk.com/dev/first_guide
- https://vk.com/dev/bots_docs

Есть ряд статей, рассказывающих как создать своего чат-бота для ВКонтакте:

- https://habr.com/ru/post/427691/
- https://habr.com/ru/post/428507/
- https://habr.com/ru/post/335106/

Есть готовая библиотека для работы с VK API:

- https://github.com/python273/vk_api
- https://vk-api.readthedocs.io/en/latest/

Материалы для самостоятельного изучения

О дальнейшей работе с ботом можно посмотреть в статье, которая бралась за основу этого материала:

https://tproger.ru/translations/telegram-bot-create-and-deploy/ Деплоить бот не нужно.

Также за основу взята вот эта подробная инструкция по ботам: https://core.telegram.org/bots

В этой лабораторной мы разобрали самостоятельную работу с ботом, однако также существует библиотека-обёртка для работы с Bot API, которая значительно упрощает работу с API, защищает нас от допущения множества ошибок при работе с ботом и привносит дополнительный полезный функционал: https://github.com/python-telegram-bot/python-telegram-bot

Ещё статьи на эту тему:

- https://habr.com/ru/post/448310/ (здесь описана другая библиотека, упрощающая работу с ботами в Telegram)
- https://www.freecodecamp.org/news/learn-to-build-your-first-bot-in-telegram-with-python-4c99526765e4/

Таким образом, по аналогии, вы можете писать чат-боты и для других мессенджеров или платформ: для ВК, для Viber, Facebook Messenger и других. Логика работы бота сохраняется, изменяется только способ создания чат-бота на платформе и способы (или даже только адреса) запросов обновлений и отправки сообщений.