**Урок 2**

**Установка модуля telebot.   
Создание «Эхо-бота»**

На предыдущем уроке мы разобрались, кто такие **боты** и зачем они нужны. При помощи **@BotFather** создали нашего первого бота — **@SuperBotick\_bot**. Изучили как при помощи команд провести первоначальную настройку бота, задать аватарку. Заполнили описание бота и приветственное сообщение.

Пришло время установить **Python** (язык программирования, на котором мы будем писать нашего бота), и среду разработки, например, **PyCharm** (можно использовать другую, в том числе онлайн-среду **Replit**). Также нам понадобится библиотека, которая поможет создать телеграм-бота.

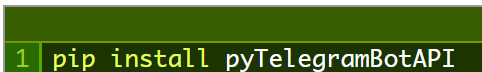
**Библиотеки для создания бота.**

Для создания телеграм-ботов на Python существует несколько десятков библиотек. Они различаются популярностью, функциональностью. Рассмотрим самые популярные.

* **Aiogram.** Современная библиотека, набирающая популярность: многие чат-боты написаны на ней. Библиотека реализует асинхронное выполнение кода, что позволяет не останавливать работу бота в ожидании ответа пользователя.
* **Python-telegram-bot**. Одна из первых библиотек для создания ботов. Отличается от Aiogram синхронным подходом к работе, то есть при ожидании ответа от пользователя выполнение кода останавливается.
* **TeleBot**. Библиотека для создания простых ботов, позволяющая работать с асинхронным и синхронным подходом на выбор. Подходит для небольших проектов.

Мы начнём создавать программу с помощью библиотеки **TeleBot.** Чтобы добавить её в наш проект необходимо открыть терминал (или командную строку) и выполнить следующую команду:

pip install pyTelegramBotAPI



Мы обращаемся к **pip** и хотим установить (**install**) модуль **pyTelegramBotAPI**, который и отвечает за создание и работу ботов.

Если модуль успешно установился, то мы должны увидеть надпись Successfully installed pyTelegramBotAPI-x.x.x.

Простыми словами **pyTelegramBotAPI** — это такой «конструктор», в котором содержатся все необходимые нам компоненты-«кубики».

**Начинаем писать код**

Можно переходить к самой интересной части — написанию кода нашего бота.

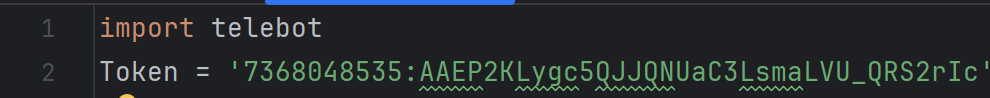
Подключаем модуль **telebot**, который мы установили ранее. Делаем это так:



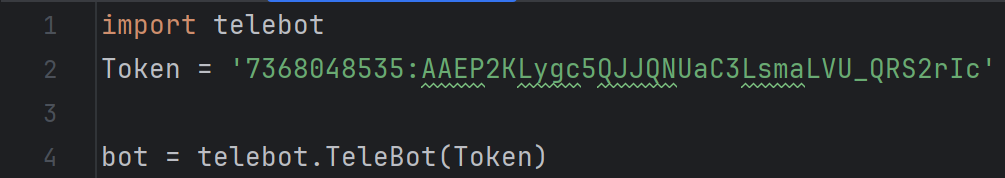
В первой части при создании бота, мы получили **Token бота**. Это ключ, с помощью которого мы сможем управлять нашим роботом через Python.

Добавляем следующую строку, где присваиваем переменной Token наш ключ-строку:

Token = 'здесь\_должен\_быть\_ключ' # полученный у @BotFather



Теперь можно создать объект нашего робота, так и назовём его — **bot**. В аргументах указываем ключ для аутентификации в Telegram:

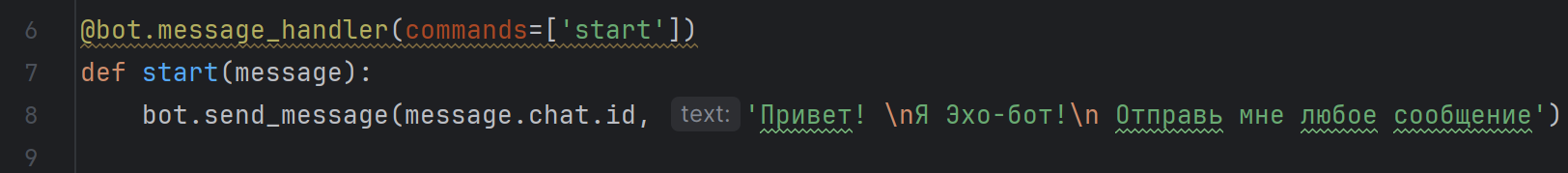


Отлично, мы передали его управление Python.

А сейчас давайте подумаем, что будет делать наш робот. Поскольку мы только учимся, не стоит делать нашего бота очень сложным.

Когда кто-то запускает бота, то бот здоровается и предлагает пользователю что-то написать. Далее бот реагирует на каждое сообщение — отвечает на сообщение пользователя его же словами.

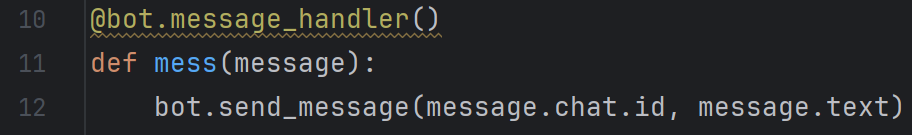
**Учим бота говорить**

Библиотека **pyTelegramBotAPI** (модуль для Python) поддерживает **декораторы** (или хэндлеры), с помощью которых можно научить бота отвечать на различные сообщения. Подробнее об обработчиках можно почитать в Telegram Bot API. Создадим декоратор, который будет отвечать на команду /start (эта команда соответствует нажатию кнопки Start в профиле бота в Telegram); внутри обработчика создадим функцию, в которой бот будет здороваться с пользователем.

Давайте разберемся в коде:

* **message\_handler** — это декоратор, который реагирует на входящие сообщения и содержит в себе функцию ответа;
* **commands=['start']** — это команда, которая связана с декоратором и запускает вложенную в него функцию;
* **def start**— создаёт функцию, которая принимает в себя сообщение пользователя **message**. Саму функцию можно назвать любым образом. Мы выбрали start, чтобы название было понятным и осмысленным;
* **bot.send\_message** — с помощью этой команды бот отправляет сообщение. Первый аргумент функции **send\_message** — уникальный ID пользователя Telegram (получаем его из предыдущего сообщения **message.chat.id**), который обратился к нашему боту, и которому нужно отправить сообщение; второй аргумент — строка, текст сообщения.

Теперь создадим событие, которое будет обрабатывать введённое пользователем сообщение:



Так как бот должен реагировать на любое текстовое сообщение от пользователя, то скобки в **@bot.message\_handler** мы оставляем пустыми.

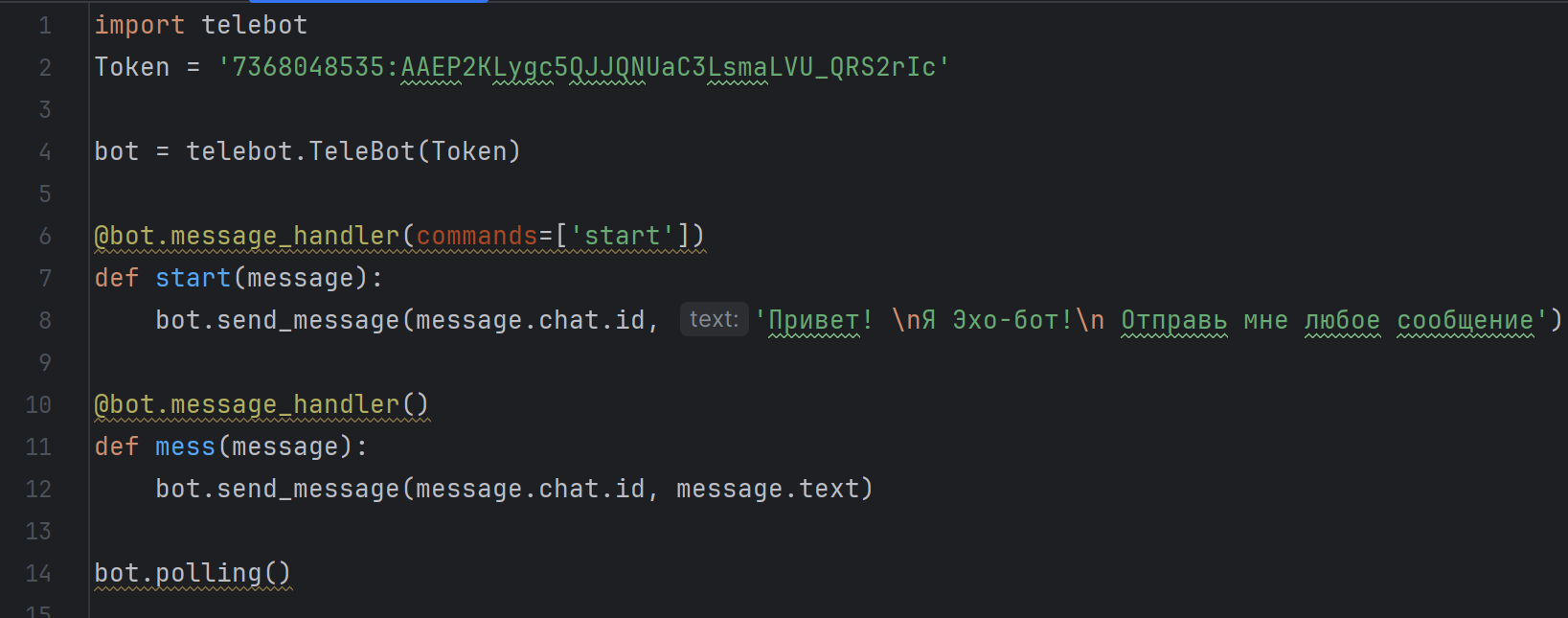
Для ответа мы также используем метод **message**, указывая, что возвращаем исходный текст, принятый в **message**.

Вот так с помощью двух функций мы научили нашего робота несложным действиям.

Последнее, что осталось добавить в код:



Можно запустить бота и без этой строки, но тогда он проработает не более одной секунды. При помощи функции **polling** наш робот постоянно обращается к серверам Telegram и как бы спрашивает «есть ли новые сообщения для меня?», получая ответы от Telegram.

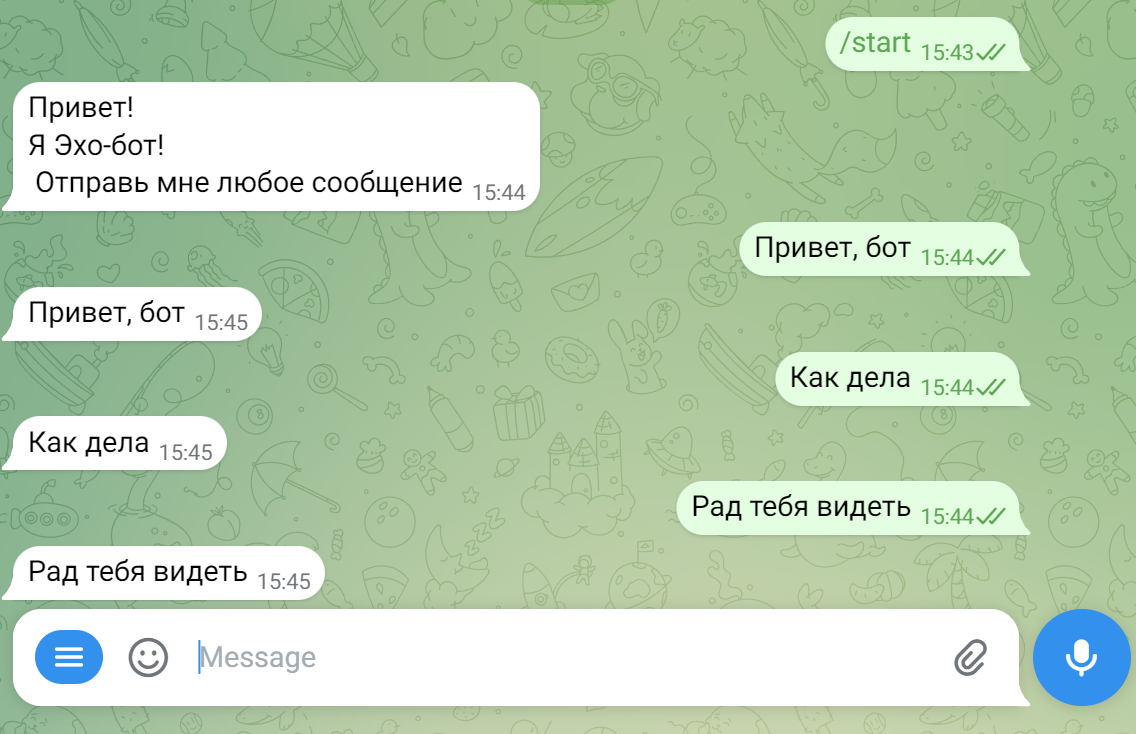
Весь код нашего бота выглядит так:

**Самостоятельная работа**

Запустите наш код и проверьте что произойдёт.

Ничего не происходит? Замечательно, всё так и должно быть. Может даже показаться, что терминал завис, но это совсем не так.

Не закрываем терминал и открываем нашего бота в Telegram. Бота можно найти по логину, у нас был @SuperBotick\_bot, либо по ссылке из сообщения от @BotFather. Запускам бота кнопкой Start.



Как мы видим — всё работает. Бот возвращает нам наши сообщения.

**Самостоятельная работа**

Добавьте ещё одну команду **bot.send\_message**, чтобы бот сказал ещё что-то.

Что мы будем делать дальше?

Расширять функциональность бота, указывая для разных команд пользователя разные ответы. Например, добавить раздел помощи, который будет появляться по команде /help.

**Рефлексия**

* Сегодня мы научились устанавливать дополнительные библиотеки
* Познакомились с библиотекой telebot
* Создали первую программу для нашего бота
* Узнали, что такое декораторы и как можно их использовать при работе с telebot
* Создали «Эхо-бота» и протестировали его

***Тайминг.***

|  |  |  |
| --- | --- | --- |
|  | Раздел урока | Время с начала урока, мин. |
|  | Повторение понятий бот, команды, постановка цели урока | 0 |
|  | Библиотеки для работы с Telegram | 10 |
|  | Привязываем бота к программе | 20 |
|  | Учим бота говорить | 40 |
|  | Самостоятельная работа | 60 |
|  | Бескомпьютерная деятельность | 65 |
|  | Самостоятельная работа | 70 |
|  | Рефлексия | 85 |