Введение, простой echo-бот

Приветствую тебя, читатель! Telegram Bot API – это мощный инструмент для вообще чего угодно. Автоматизация действий, работа с пользователями, онлайн-магазины, игры и много чего ещё. В этом учебнике мы научимся писать ботов для Telegram на языке Python.

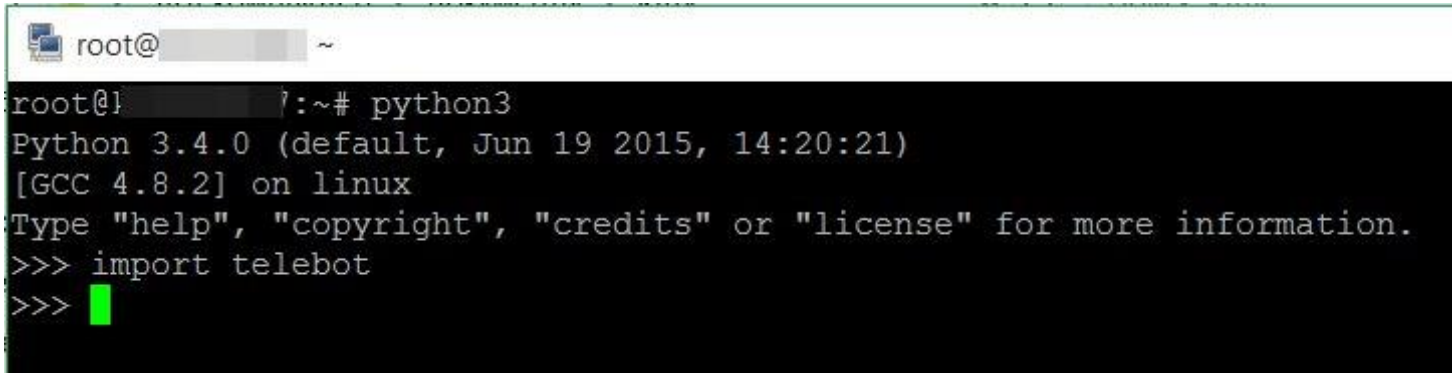
Сразу оговорюсь: тот бот, который получится в итоге - это лишь прототип, цель всех этих постов - рассказать об основах ботостроения, показать, как можно за короткое время написать простого бота для своих нужд.

Язык программирования будет Python 3, но это не означает, что любители PHP, Ruby и т.д. в пролёте; все основные принципы совпадают. Я не буду особо останавливаться на описании самого языка, желающие могут ознакомиться с документацией по Python здесь.

Подготовка к запуску

Взаимодействие ботов с людьми основано на HTTP-запросах. Чтобы не мучаться с обработкой «сырых» данных, воспользуемся библиотекой pyTelegramBotAPI, которая берет на себя все нюансы отправки и получения запросов, позволяя сконцентрироваться непосредственно на логике. Установка библиотеки предельно простая:

```
pip install pytelegrambotapi
python3
```



Скриншот из терминала с запущенным интерпретатором Python

> Обратите внимание: библиотека называется `pyTelegramBotAPI`, а не `telebot`. Последнюю ставить **не нужно**!

Теперь можно выйти из режима Python-консоли (Ctrl+Z или Ctrl+D, или `exit()`)

Пишем простого echo-бота

Ну, довольно слов, перейдем к делу. В качестве практики к первому уроку, напишем бота, повторяющего присланное текстовое сообщение. Создадим каталог, а внутри него 2 файла: `bot.py` и `config.py`. Я рекомендую выносить различные константы и настройки в файл `config.py`, дабы не загромождать другие. В файл `config.py` впишем:

```
# Токен ненастоящий :) Подставьте свой
token = '1234567890:AAE_abCDEFghijKLmNOpqRsTuVWxyz'
```

Теперь надо научить бота реагировать на сообщения. Напишем обработчик, который будет реагировать на все текстовые сообщения.

```
@bot.message_handler(content_types=["text"])
def repeat_all_messages(message): # Название функции не играет никакой роли
    bot.send_message(message.chat.id, message.text)
```

У читателя возникнет вопрос: зачем там символ “@”? Что это вообще за `message_handler`? Дело в том, что после приёма сообщения от Telegram его надо обработать по-разному в зависимости от того, что это за сообщение: текст “привет” или текст “пока”, может быть, вообще стикер или музыка. Первое, что придёт в голову – написать множество конструкций `if-then-else`, но такой подход некрасивый и позволяет быстро запутаться.
Для решения этой проблемы автор библиотеки pyTelegramBotAPI реализовал механизм хэндлеров, которые используют питоновские декораторы (пока просто запомним это слово). В хэндлере описывается, в каком случае необходимо выполнять ту или иную функцию. Например, хэндлер `@bot.message_handler(content_types=["text"])` выполнит нижестоящую функцию, если от Telegram придёт текстовое сообщение, а хэндлер `@bot.message_handler(commands=["start"])` сработает при получении команды **/start**.

Теперь запустим бесконечный цикл получения новых записей со стороны Telegram:

```
if __name__ == '__main__':
    bot.infinity_polling()
```

Функция infinity_polling запускает т.н. Long Polling, бот должен стараться не прекращать работу при возникновении каких-либо ошибок. При этом, само собой, за ботом нужно следить, ибо сервера Telegram периодически перестают отвечать на запросы или делают это с большой задержкой приводя к ошибкам 5xx)

Итак, полный код файла `bot.py` выглядит следующим образом:

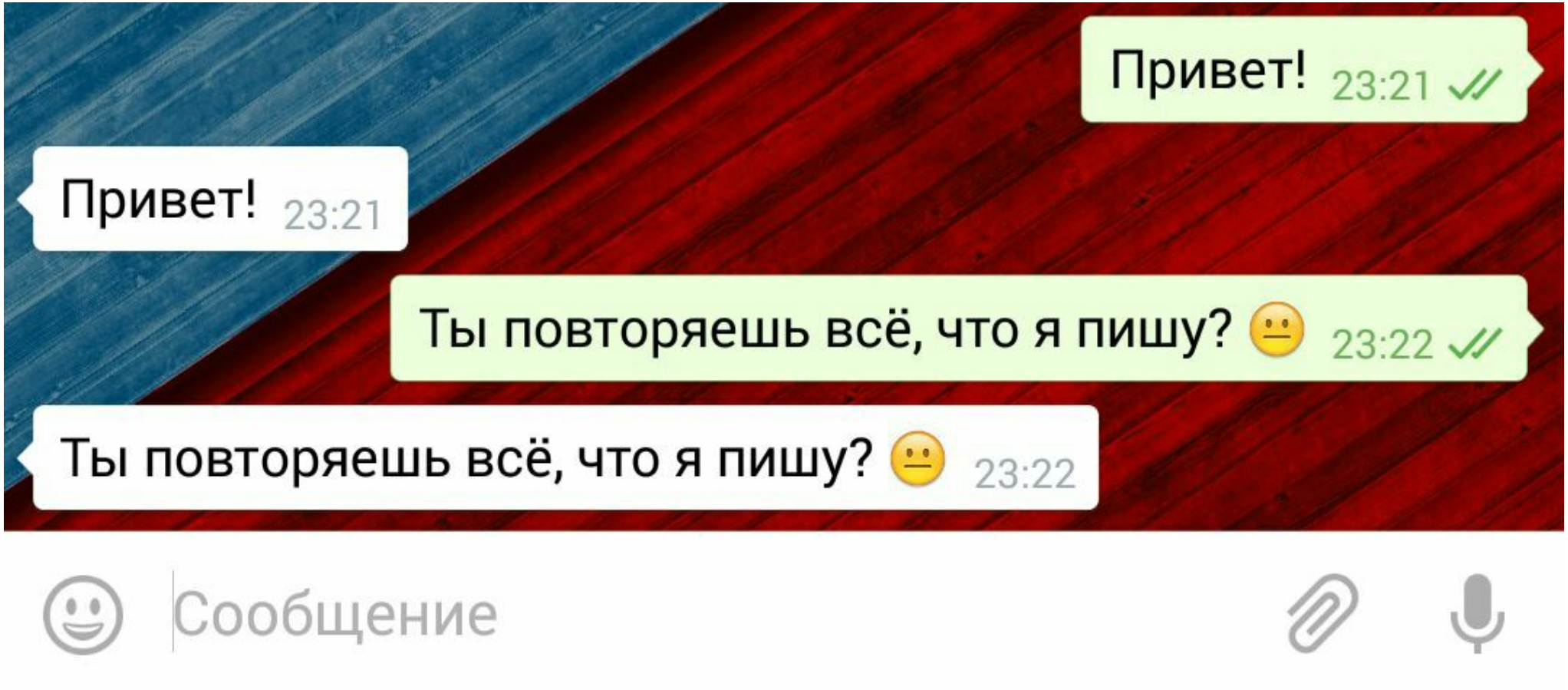
```
import config
import telebot

bot = telebot.TeleBot(config.token)

@bot.message_handler(content_types=["text"])
def repeat_all_messages(message): # Название функции не играет никакой роли
    bot.send_message(message.chat.id, message.text)

if __name__ == '__main__':
     bot.infinity_polling()
```

Готово! Осталось запустить бота: `python3 bot.py`



Бот работает

На этом первый урок окончен.

Урок №2 →