Программирование на языке Python

Тема №6
Голосовые помощники и чат боты

Занятие 1	Занятие 2								

Список используемой литературы.

1. Программирование для детей на языке Python – Москва: Издательство АСТ – 2017.
2. Python для детей. Самоучитель по программированию / Джейсон Бриггс (перевод С. Ломакина) – 2017.
3. Сам себе программист. Как научиться программировать и устроиться в Ebay? / Кори Альтхофф – Москва: Эксмо, 2018.
4. Изучаем Python. том 1. 5-е изд. – СПб. : ООО «Диалектика», 2019.
5. Изучаем Python. том 2. 5-е изд. – СПб. : ООО «Диалектика», 2019.

В презентациях занятий ссылки на материал из этих книг будут обозначаться следующим образом:

[2] с.35 – это означает, что книга №2 из списка, страница 35.

То есть число в квадратных скобках – номер книги в списке литературы.

План занятия

Чат боты

- Начало работы с асинхронными функциями в Python.
- Создание простого Discord бота с помощью библиотеки discord.py
- Собственно пишем бота
- Добавляем бота на сервер

Используемая литература:

from internet messages.

ЧАТ боты.

Одним из примеров чат-бота можно считать IT-cube BOT, который успешно работает на нашем сервере. Он написан с применением библиотеки discord.py

1

Начало работы с асинхронными функциями в Python.

Вы слышали об асинхронном программировании на Python? Вам интересно узнать больше об асинхронных функциях Python и о том, как вы можете использовать их в своей работе? На занятии мы ознакомимся с применением асинхронных функций Python, а библиотека discord.py – это именно библиотека асинхронных функций.

Понимание асинхронного программирования

Синхронная программа выполняется по одному шагу за раз. Даже с условными переходами, циклами и вызовами функций вы все равно можете думать о коде с точки зрения выполнения одного шага выполнения за раз. Когда каждый шаг завершен, программа переходит к следующему.
Вот два примера программ, которые работают таким образом:

- Программы пакетной обработки часто создаются как синхронные программы. Вы получаете некоторую информацию, обрабатываете ее и создаете некоторую информацию. Шаги следуют один за другим, пока программа не достигнет желаемого результата.
- Программы командной строки — это небольшие быстрые процессы, которые выполняются в терминале. Эти сценарии используются для создания чего-либо, преобразования одной вещи во что-то другое, создания отчета или, возможно, вывода списка данных. Это последовательность шагов программы, которые выполняются последовательно, пока программа не будет завершена.

Асинхронная программа ведет себя по-другому.

Они все еще выполняются шаг за шагом. Разница в том, что система может не дождаться завершения шага выполнения, прежде чем перейти к следующему. Это означает, что программа перейдет к следующим шагам выполнения, даже если предыдущий шаг еще не завершен и все еще выполняется в другом месте. Это также означает, что программа знает, что делать после завершения предыдущего шага.

Зачем нужно писать такие программы?

Создание синхронного веб-сервера

Основная задача веб-сервера более или менее аналогична пакетной обработке. Сервер получает некоторый ввод, обработает его и создаст вывод. Написанный как синхронная программа, может работать как обычный веб-сервер. Но это также был бы ***абсолютно ужасный веб-сервер***.

Почему? В этом случае одна единица работы (ввод, обработка, вывод) — не единственная цель. Настоящая цель — как можно быстрее выполнить сотни или даже тысячи единиц работы. Это может происходить в течение длительных периодов времени, и несколько рабочих единиц могут даже прибыть все сразу.

Методы асинхронного программирования позволяют вашим программам использовать преимущества относительно медленных процессов ввода-вывода, освобождая ЦП для выполнения другой работы.

Иной взгляд на программирование

Когда вы начнете пытаться понять асинхронное программирование, вы можете увидеть много дискуссий о важности блокирования или написания неблокирующего кода. Что такое неблокирующий код? Что такое код блокировки? Могут ли ответы на эти вопросы помочь вам написать лучший веб-сервер? Если так, как это можно сделать? Давайте разберемся!

Написание асинхронных программ требует, чтобы вы по-другому относились к программированию. Хотя это новое мышление сразу может показаться сложным, но это также интересное упражнение. Представьте себе, что вы — родитель, пытающийся сделать несколько вещей одновременно. Вы должны подсчитать последние траты по чековой книжке, постирать белье и присматривать за детьми.

Давайте разберемся с этим:

- Подсчет трат по чековой книжки — синхронная задача. Один шаг следует за другим, пока не будет сделано. Вы делаете всю работу самостоятельно.
- Однако вы можете оторваться от чековой книжки, чтобы заняться стиркой. Вы выгружаете сушилку, перемещаете одежду из стиральной машины в сушилку и запускаете другую загрузку в стиральную машину.
- Работа со стиральной машиной и сушилкой является синхронной задачей, но основная часть работы происходит после запуска стиральной машины и сушилки. Как только вы их запустите, вы можете уйти и вернуться к чековой книжки.
- Следить за своими детьми — еще одна асинхронная задача. После того, как им дано какое ли задание или они начали играть, они могут делать это самостоятельно большую часть времени. Это меняется, когда кто-то нуждается во внимании, например, когда кто-то проголодался или поранился. Дети — долгосрочное задание с высоким приоритетом.

Эти примеры могут помочь проиллюстрировать понятия блокирующего и неблокирующего кода. Давайте подумаем об этом в терминах программирования. В этом примере вы похожи на процессор. Пока вы перемещаете белье, вы (процессор) заняты и не можете выполнять другую работу, например, подсчет чековой книжки. Но это нормально, потому что задача относительно быстрая.

С другой стороны, запуск стиральной машины и сушилки не мешает вам выполнять другие задачи. Это асинхронная функция, потому что вам не нужно ждать ее завершения. Как только это началось, вы можете вернуться к чему-то другому. Это называется переключением контекста (context switch): изменился контекст того, что вы делаете, и зуммер машины сообщит вам о будущем, когда задача стирки будет завершена.

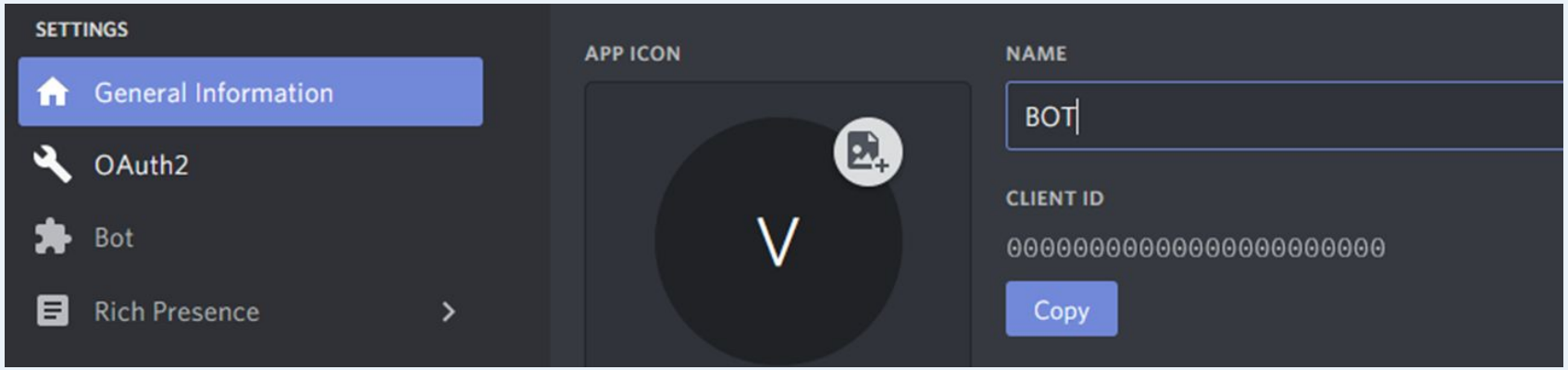
Как человек, вы, естественно, можете манипулировать несколькими вещами одновременно, часто не задумываясь об этом. Как разработчик, уловка состоит в том, как преобразовать такое поведение в код, который делает то же самое.

Создание простого Discord бота с помощью библиотеки discord.py

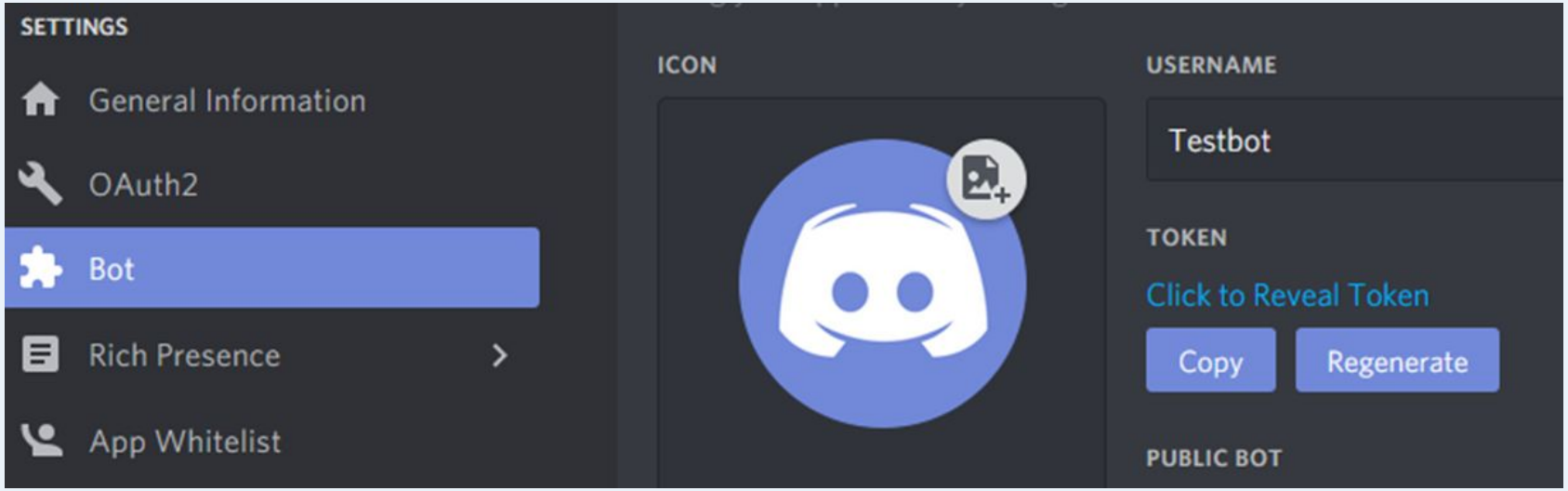
Асинхронная библиотека discord.py содержит все что нужно для бота, с помощью нее даже можно работать с голосовыми каналами сервера. В этом разделе я расскажу как создать простенького бота для вашего discord сервера.

Получение токена и Client ID для вашего бота

Для получения токена и ID бота небходимо создать свое приложение и в разделе General Information скопировать Client ID.



А в разделе настроек создать бота и скопировать его токен. Задача не сложная, думаю все с этим справятся.



3

Собственно пишем бота.

Устанавливаем discord.py с помощью pip:

```
pip install discord
```

После успешной установки создаем файл bot.py, где будем писать бота.
Импортируем все необходимое:

```
import discord
from discord.ext import commands
```

Создаем переменную с вашим токеном, про который я писал выше:.

```
TOKEN = 'Ваш токен'
```

Создаем тело бота:

```
bot = commands.Bot(command_prefix='!') #инициализируем бота с префиксом '!'
```

Для начала сделаем простенькую команду, аргумент которой бот будет просто пересылать:

```
@bot.command(pass_context=True) #разрешаем передавать аргументы
async def test(ctx, arg): #создаем асинхронную функцию бота
    await ctx.send(arg) #отправляем обратно аргумент
```

И в конце запускаем бота с вашим токеном:

```
bot.run(TOKEN)
```

4

Добавляем бота на сервер.

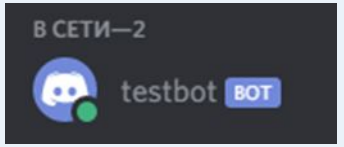
Теперь необходимо добавить бота на сервер. Сделать это можно с помощью ссылки:

```
https://discordapp.com/oauth2/authorize?&client_id={Client ID}&scope=bot&permissions={Права, например 66395456}
```

Число необходимых прав можно получить в разделе настроек бота.
Теперь можно запускать бота:

```
python bot.py
```

После нескольких секунд, можно заметить его в сети:



И наконец-то попробовать отправить команду:

Заключение

Вот так можно легко запустить у себя на сервере бота. Как можно заметить библиотека делает практически все за тебя и остается только добавлять свой функционал с использованием python. В следующий раз я покажу как следить за событиями, подключатся к голосовым каналам (избегая проблем с linux и Windows), использовать роли и права участников и другое.

Что такое бот?

https://cyberguru.tech/программирование/как-сделать-discord-bot-в-python#nine

Discord растет в популярности. Таким образом, автоматизированные процессы, такие как запрет несоответствующих пользователей и реагирование на запросы пользователей, жизненно важны для процветания и развития сообщества.

Автоматизированные программы, которые выглядят и действуют как пользователи и автоматически реагируют на события и команды в Discord, называются пользователями-ботами . Пользователи Discord ботов (или просто боты) имеют почти неограниченное количество приложений .

Например, допустим, вы управляете новой гильдией Discord, и пользователь присоединяется впервые. Вы можете лично обратиться к этому пользователю и приветствовать его в своем сообществе. Вы также можете рассказать ему о своих каналах или попросить его представиться.

Пользователь чувствует себя желанным гостем и получает удовольствие от обсуждений, происходящих в вашей гильдии, а затем он, в свою очередь, приглашает друзей.

Со временем ваше сообщество становится настолько большим, что больше не представляется возможным лично связаться с каждым новым участником, но вы все равно хотите отправить им что-нибудь, чтобы признать их новым членом гильдии.

С ботом можно автоматически реагировать на присоединение нового члена вашей гильдии. Вы даже можете настроить его поведение в зависимости от контекста и контролировать его взаимодействие с каждым новым пользователем.

Это здорово, но это только один маленький пример того, как бот может быть полезен.

Как сделать Discord Bot в Python

Поскольку вы изучаете, как создать бот Discord с Python, вы будете использовать его discord.py. Библиотека discord.py в Python, исчерпывающе и эффективно реализует API-интерфейсы Discord. Она включает в себя использование Async IO в Python реализации. Начните с установки discord.py с pip:

```
$ pip install -U discord.py
```

Теперь, когда установили discord.py, вы будете использовать эту библиотеку для создания вашего первого подключения к Discord!

Создание Discord Connection

Первым шагом в реализации вашего бота является создание подключения к Discord. С помощью discord.py вы создаете экземпляр Client:

```
# bot.py
import os
import discord
from dotenv import load_dotenv

load_dotenv()
TOKEN = os.getenv('DISCORD_TOKEN')

client = discord.Client()

@client.event
async def on_ready():
    print(f'{client.user} has connected to Discord!')

client.run(TOKEN)
```

`Client` – это объект, представляющий связь с Discord. `Client` обрабатывает события, отслеживает состояние и обычно взаимодействует с API-интерфейсами Discord.

Вы создали `Client` и реализовали `on_ready()` – обработчик событий, который обрабатывает событие, когда `Client` установил соединение с Discord и завершил подготовку данных, отправленных Discord, таких как состояние входа в систему, данные гильдии и канала и т. д.

Другими словами, `on_ready()` будет вызван (и ваше сообщение будет напечатано), как только `client` будет готов к дальнейшим действиям.

Когда вы работаете с такими секретами, как токен Discord, рекомендуется записать его в свою программу из переменной среды.

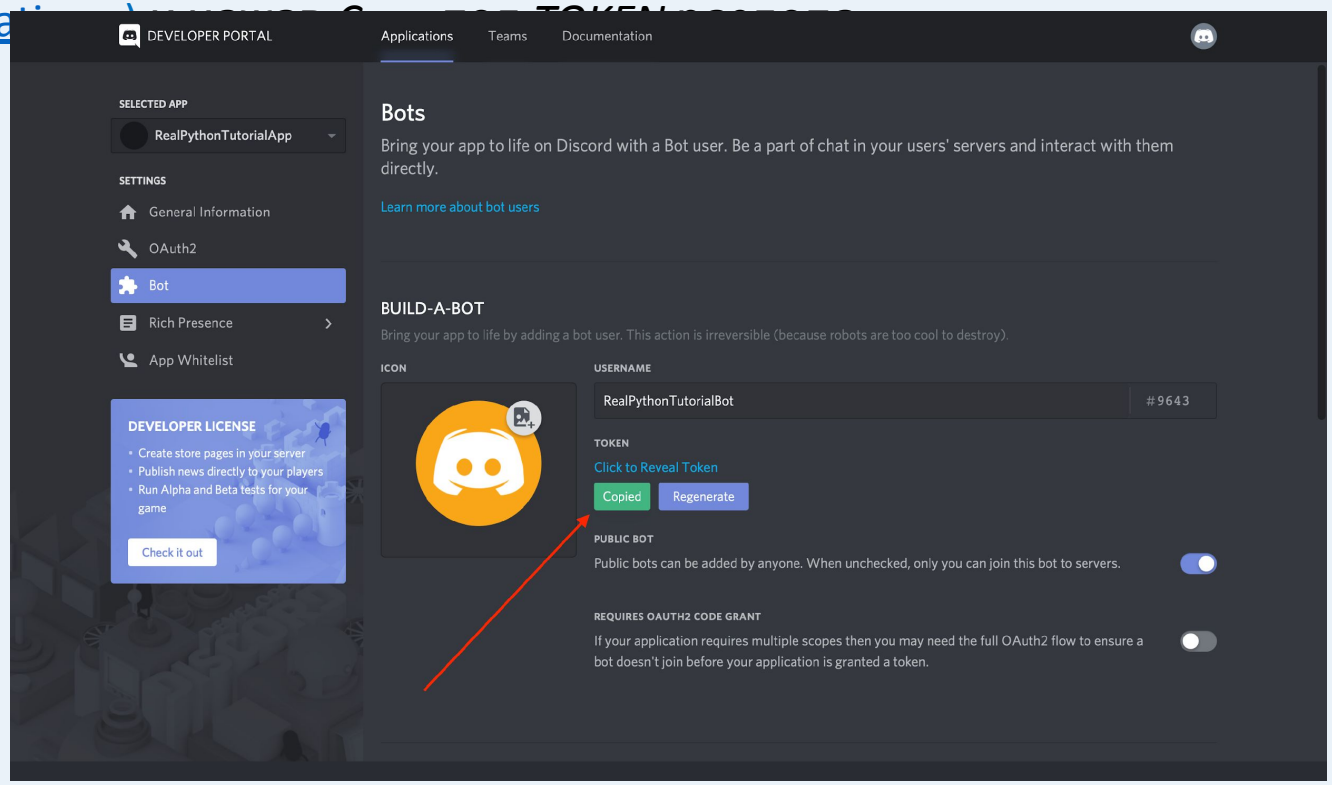
Использование переменных среды помогает вам:

- Избегайте помещения секретов в систему контроля версий
- Используйте разные переменные для среды разработки и производства без изменения кода

Хотя вы могли бы `export DISCORD_TOKEN={your-bot-token}`, более простое решение – сохранить .env файл на всех машинах, на которых будет выполняться этот код. Это не только проще, так как вам не придется export каждый раз очищать свою оболочку, но и защищает вас от сохранения ваших секретов в истории вашей оболочки.
Создайте файл с именем .envв том же каталоге, что и bot.py:

```
# .env
DISCORD_TOKEN={your-bot-token}
```

Вам нужно заменить `{your-bot-token}` с ботами маркером, который вы можете получить, перейдя обратно на Bot страницу на Developer Portal (ссылка: http://discordapp.com/developers/applica[illegible] ТОКЕН вашего



Оглядываясь на bot.py код, вы заметите библиотеку под названием dotenv. Эта библиотека удобна для работы с .env файлами. load_dotenv() загружает переменные окружения из .env файла в переменные окружения вашей оболочки, чтобы вы могли использовать их в своем коде.

Установить dotenv с помощью pip:

```
$ pip install -U python-dotenv
```

Наконец, `client.run()` запускается ваш Client токен вашего бота.
Теперь, когда вы настроили оба bot.py и .env, вы можете запустить свой код:

```
$ python bot.py
RealPythonTutorialBot#9643 has connected to Discord!
```

Вы подключились с помощью Client к Discord, используя токен вашего бота.