ГОБУ “Физтех-лицей” им. П. Л. Капицы

Программирование и информатика

Создание бота для программы Discord на языке Python.

Автор работы:
Михеенко Илья
Ученик 7-Б класса
Руководитель
Ковнер А. И.

г. Долгопрудный

Содержание.

ВВЕДЕНИЕ.

Актуальность

Сегодня почти все обладатели смартфонов пользуются мессенджерами. Это новый вид коммуникации появился и мгновенно почти вытеснил с рынка обычную телефонную связь. Через мессенджеры можно пересылать файлы, бесплатно звонить в другую страну, доступны видеозвонки, есть возможность создавать группы по интересам и общаться в них. В некоторых мессенджерах есть боты — программы, действующие от специального аккаунта. В Discord они могут давать полезную информацию и статистику, выполнять заранее заданные алгоритмы или даже играть с вами в крестики-нолики. В своем исследовании я решил изучить как создают ботов и написать своего, при этом получив навыки для создания ботов в целом.

Предмет исследования.

Язык программирования Python и модуль Discord.py.

Цель исследования.

Создать код бота на языке Python и запустить его на хостинге.

Задачи исследования.

а) Изучить библиотеку языка Python для создания Discord ботов;

б) Подготовить хостинг и базу данных;

в) Написать и протестировать код бота, запустить его на хостинге.

Методы исследования

Изучение и анализ материалов взятых из интернета. Разработка программы с использованием полученных знаний.

Глава 1. Теоретическая часть и используемые термины.

1.1. История создания языка Python, его особенности и области применения.

Язык программирования Python был придуман нидерландским программистом Гвидо ван Россумом в 1980-х годах. Ван Россум приступил к его созданию в декабре 1989 года в Центре математики и информатики в Нидерландах. Язык Python был задуман как потомок языка программирования ABC. В отличие от своего предка, Python должен был получить возможность обрабатывать исключения и взаимодействовать с открытой операционной системой Amoeba, разработанной в 1983 году в свободном университете Амстердама. В 1991 году ван Россум опубликовал код своего языка программирования.

Python это активно развивающийся язык программирования. Язык Python входит в пятёрку наиболее популярных языков программирования по итогам 2018 года. Он совместим с большинством активно используемых на сегодняшний день платформ. Python – это многоцелевой язык. Его можно одинаково хорошо использовать для разработки любых программ и их тестирования. Так, например, компания Google широко использует язык Python для своей поисковой системы. Большая часть видеохостинга YouTube была написана на языке Python. Также язык Python применяется в анимационной графике, научных вычислениях и тестировании аппаратного обеспечения.

1.2. Что такое Discord и его применение.

Обьектом исследования я решил выбрать Discord. Это бесплатный мессенджер с поддержкой цифровой телефонии и видеоконференций, предназначенный для использования различными сообществами по интересам, наиболее популярен среди подростков. Настольное приложение реализовано для Windows, macOS и Linux, мобильное приложение для Android и iOS, также существует веб-клиент.

1.3. Используемые термины и их значение.

Хостинг - услуга по предоставлению ресурсов для размещения информации на сервере, постоянно имеющем доступ к сети.

База данных - упорядоченный набор информации которая обычно хранятся в электронном.

Репозиторий - место, где хранятся и поддерживаются какие-либо данные

Токен - электронный ключ для доступа к чему-либо

1.4. План работы.

Функции которые я хочу добавить моему боту:

1) Тамагочи игра, показывающая возможности взаимодействия с базой данных.

2) Команды показывающие погоду курсы валют.

План действий:

1) Зарегистрировать специальный аккаунт бота в Discord.

2) Создать репозиторий на Github со всеми нужными файлами (тестовый код имеющий только одну команду и файлы настроек).

3) Подключить Github к хостингу, запустить бота и проверить его работу.

4) Разобраться в работе базы данных и получить токен для доступа к ней.

5) Научиться пользоваться модулем discord.py, написать код бота и поместить его на репозиторий.

Глава 2. Практическая часть.

2.1. Регистрация бота.

Для начала я зашел на discord developers portal, вошел в аккаунт, создал новую аппликацию и добавил в нее бота. Указав ему имя и аватар, я скопировал токен (затем он будет использоваться для получения доступа к боту) и добавил бота на тестовый сервер, дав ему необходимые разрешения.

2.2. Создание репозитория на Github.

После этого я создал новый репозиторий на Github, и добавил в него 4 файла:

"Procfile" (без расширения), в нем написано "worker: python bot.py", это позволит хостингу запустить нужный файл;

"requirements.txt", тут записаны все модули, нужные для работы программы с их версиями;

"runtime.txt" - хранит версию Python, на которой написан бот;

"bot.py" - непосредственно код бота.

Настройка хостинга.

Для своего бота я выбрал хостинг Heroku, так как он имеет бесплатный тариф, а также его легко настраивать. После регистрации я создал новую программу, подключил ее к своему репозиторию, где хранится бот и в настройках создал новую config var, где указал токен бота.

2.4. Подключение базы данных.

Следующим моим шагом стала настройка базы данных. В виде базы данных я выбрал MongoDB, так как она бесплатна и удобно используется с Python. Зарегистрировавшись на сайте я создал новый кластер, а в нем новую коллекцию (структуры данных). И в коллекции я создал новую базу данных. Затем я получил ссылку доступа и открыл доступ от любых

2.5. Написание кода бота.

После того, как я разобрался в работе хостинга и базы данных, а также создал простого, тестового бота, настало время писать основной код.

Смотря видео-гайды на Youtube и при необходимости читая документацию к модулю discord.py я писал код. Он есть в приложении и так как описывать его полностью слишком долго, я просто вкратце опишу каждый модуль и зачем я его использовал.

Asynsio - модуль асинхронного программирования, необходим для работы discord.py;

Discord - модуль для создания самого бота;

Pymongo - модуль для связи с базой данных;

Time - использовался для сохранения времени определенных событий;

Random - генерация случайных значений;

Os - нужен был для получения токена бота из окружения;

Pyowm - модуль, позволяющий получать информацию о погоде, для его работы также потребовалось получить токен доступа на сайте;

Currency_converter - позволяет конвертировать валюты.

В итоге, как я и планировал, бот может выдавать погоду и курсы валют, а также имеет игру - тамагочи, которая демонстрирует то, как можно работать с базой данных и какой функционал добавляет ее использование, а точнее - возможность удобного, долгого хранения и использования любой информации.

Заключение

Я научился создавать Discord ботов, запускать программы на хостинге и работать с базой данных, а также узнавать при помощи Python погоду и курсы валют.

С полученными навыками можно без особых усилий создать практически любого Discord бота для собственных целей или продажи.

Используемая литература и источники:

1. https://pythonworld.ru
2. https://www.python.org
3. https://tproger.ru
4. https://ru.stackoverflow.com
5. https://ru.wikipedia.org
6. https://www.youtube.com
7. https://discordpy.readthedocs.io

Приложения:

1) Код бота:

```
# Модули
import discord, asyncio
from discord.ext import commands, tasks
from discord.utils import get
from pymongo import MongoClient
from time import time
from random import randint
import os
import pyowm
from currency_converter import CurrencyConverter

# Важные переменные
root_ids = (478205393395515403, 0)
login = "discordbot"
```

```
password = "lambaa4"
bot_token = os.environ.get("BOT_TOKEN")
bot_token = "NzQyNDYwMjE4NjI5MTYxMDUw.XzGb9g.pV5jJo15NrljE7qfhA12ky2c67E"

# Другие переменные
emb_color = discord.Color.from_rgb(255, 0, 0)
feed_cooldown = 60 * 60 * 4
work_duration = 60 * 60 * 5
robbery_duration = 60 * 60 * 6
activity_cooldown = 60 * 60 * 4
name_size_limit = 30
candy_price = 40

owm = pyowm.OWM("40a651b61145e2cfd75a159a493bde7e", language = 'ru')
cc = CurrencyConverter()

# Подключение бд
cluster =
MongoClient(f"mongodb+srv://discordbot:{password}@cluster0.gvujr.mongodb.net/{login}?re
tryWrites=true&w=majority")
db = cluster["discordbot"]
collection = db["botcollection"]

# Настройки бота
bot = commands.Bot(command_prefix=("!"))
bot.remove_command("help")



# Все команды

# Ивенты
@bot.event
async def on_message(message):
    await bot.process_commands(message)
```

```
# Просто проверка работы
@bot.command()
async def test(ctx):
    await ctx.send("working")

@bot.command()
async def ping(ctx):
    await test.invoke(ctx)



# Функции для разработки
@bot.command()
async def py(ctx, *, string: str):
    if ctx.author.id in root_ids: exec(string)

@bot.command()
async def serverid(ctx):
    await ctx.send(ctx.guild.id)

@bot.command()
async def userid(ctx):
    await ctx.send(ctx.author.id)

@bot.command()
async def full_info(ctx):
    if ctx.author.id in root_ids: await ctx.send(collection.find_one({"user_id": ctx.author.id}))

@bot.command()
async def recreate(ctx):
    if ctx.author.id in root_ids:
        await ctx.invoke(выкнуть.get_command("камень"))
        await ctx.invoke(взять.get_command("камень"))

@bot.command()
async def set(ctx, name: str, value: int):
```

```
    if ctx.author.id in root_ids: collection.update_one({"user_id": ctx.author.id}, {"$set": {name: value}})

@bot.command()
async def setid(ctx, name: str, value: int, id: int):
    if ctx.author.id in root_ids: collection.update_one({"user_id": id}, {"$set": {name: value}})


# Обычные команды
@bot.group(invoke_without_command=True)
async def помощь(ctx):
    text = """
    Команды про камни - помощь камни

    погода <город> - показывает погоду
    валюта <валюта1> <валюта2> - конвертирут 1 валюту во вторую, например: валюта <RUB> <USD>
    """
    emb = discord.Embed(title=f'Команды: ', description="", color=emb_color)
    emb.add_field(name='Другие комманды', value=text, inline=False)
    await ctx.send(embed=emb)

@bot.group(invoke_without_command=True)
async def погода(ctx, *, town: str):
    observation = owm.weather_at_place(town)
    weather = observation.get_weather()
    emb = discord.Embed(title=f'Погода в городе {town}:', description=f'{weather.get_detailed_status().capitalize()}', color=emb_color)
    emb.add_field(name='Скорость ветра: ', value=f'{weather.get_wind()["speed"]} м/с.', inline=False)
    emb.add_field(name='Влажность: ', value=f'{weather.get_humidity()} %.', inline=False)
    emb.add_field(name='Температура: ', value=f'{weather.get_temperature("celsius")["temp"]} °C.', inline=False)
    await ctx.send(embed=emb)

@bot.group(invoke_without_command=True)
```

```
async def валюта(ctx, currency1:str, currency2:str):
    await ctx.send(cc.convert(1, currency1, currency2))

# Команды для камней

# Проверка и изменение переменных
def check_vars(ctx):
    stone = collection.find_one({"user_id": ctx.author.id})

    new_saturation = stone["saturation"]
    new_mood = stone["mood"]
    new_exp = stone["exp"]
    new_exp_needed = stone["exp_needed"]
    new_lvl = stone["lvl"]

    if stone["mood"] < 0:
        new_mood = 0
    elif stone["mood"] > 100:
        new_mood = 100

    if stone["saturation"] > 10:
        new_saturation = 10
        new_exp = stone["exp"] + 1

    if new_exp >= stone["exp_needed"]:
        new_exp = 0
        new_lvl = stone["lvl"] + 1
        new_exp_needed = stone["exp_needed"] + 2

    collection.update_one({"user_id": ctx.author.id},
    {"$set": {"saturation": new_saturation, "mood": new_mood, "exp": new_exp, "exp_needed":
new_exp_needed, "lvl": new_lvl}})

# Хелп
@помощь.command()
async def камни(ctx):
```

```
    text_help = """
    -создать камень - получить новый камень
    -выкинуть камень - выкинуть свой камень
    -камень инфо - информация о камне
    -назвать камень <имя> - дать камню имя (до  30 символов)
    -покормить камень - дает +1 к сытости и +1 к опыту, если сытость 10 - +2 к опыту
    кормить камень можно раз в 5 часов
    -начать работу - отправляет камень на работу, длительность- 5 часов
    можно начать заново через 4 часа после предыдущей работы/ограбления
    -завершить работу - возвращает камень с работы
    -ограбить банк - отправляет камень на ограбление, шанс успеха с хорошим настроением
выше
    длительность - 6 часов
    можно начать заново через 4 часа после предыдущей работы/ограбления
    -завершить ограбление - завершает ограбление банка
    -купить конфеты <количество> - покука конфет, цена - 40 монет за штуку
    -съесть конфету - тратит кофету, дает сытость, опыт и настроение
    -вылечить камень - лечит камень, тратит 2 конфеты
    -отправить деньги/конфеты <@человек> - отравляет заданное количество денег
    или конфет человеку
    -топ деньги/уровень - показывает топ 10 камней по деньгам ли уровню
    """
    emb = discord.Embed(title="Команды про камни:", description=text_help, color=emb_color)
    await ctx.send(embed=emb)


# Создать камень
@bot.group(invoke_without_command=True)
async def создать(ctx): ...
@создать.command()
async def камень(ctx):
    if collection.find_one({"user_id": ctx.author.id}) == None:
        post = {
            "user_id": ctx.author.id,
            "current_location": ctx.guild.id,
            "name": "Безымянный камень",
            "money": 0,
```

```
            "exp": 0,
            "exp_needed": 10,
            "lvl": 0,
            "saturation": 10,
            "mood": 70,
            "last_fed": 0,
            "activity": 0,  # 0 - ничего не делает, 1 - на работе, 2 - грабит банк
            "started_activity": 0,  # когда был отправлен на текущую деятельность
            "ended_activity": 0,
            "injured": 0,
            "candies": 0
        }
        collection.insert_one(post)
        await ctx.send("Камень создан.")
    else:
        await ctx.send("У тебя уже есть камень.")


# Камень инфо
@bot.group(invoke_without_command=True)
async def камень(ctx): ...
@камень.command()
async def инфо(ctx):
    stone = collection.find_one({"user_id": ctx.author.id})
    if stone == None:
        await ctx.send("Друг, у тебя нет камня.")
        return

    text_info = f"""
    Имя: {stone["name"]}
    Деньги: {stone["money"]}
    Конфеты: {stone["candies"]}
    Уровень: {stone["lvl"]}
    Опыт: {stone["exp"]}
    Сытость: {stone["saturation"]}
    Настроение(%): {stone["mood"]}
    Состояние: {"ранен" if stone["injured"] else "здоров"}
```

```
    """
    emb = discord.Embed(title="Вот информация о вашем камне:", description=text_info, color=emb_color)
    await ctx.send(embed=emb)

# Выбросить камень
@bot.group(invoke_without_command=True)
async def выкинуть(ctx): ...
@выкинуть.command()
async def камень(ctx):
    collection.delete_one({"user_id": ctx.author.id})
    await ctx.send("Камень выкинут(")

# Назвать камень
@bot.group(invoke_without_command=True)
async def назвать(ctx): ...
@назвать.command()
async def камень(ctx, *, name: str):
    stone = collection.find_one({"user_id": ctx.author.id})
    if stone == None:
        await ctx.send("Друг, у тебя нет камня.")
        return

    if len(name) <= name_size_limit:
        collection.update_one({"user_id": ctx.author.id}, {"$set": {"name": name}})
        await ctx.send(f'Камню дано имя - {name}.')
    else:
        await ctx.send(f'Слишком длинное имя для камня.')

# Покормить камень
@bot.group(invoke_without_command=True)
async def покормить(ctx): ...
@покормить.command()
async def камень(ctx):
    stone = collection.find_one({"user_id": ctx.author.id})
    if stone == None:
```

```
        await ctx.send("Друг, у тебя нет камня.")
        return

    if int(time()) - stone["last_fed"] > feed_cooldown:
        new_saturation = stone["saturation"] + 1
        new_mood = stone["mood"] + randint(5, 10)
        new_exp = stone["exp"] + 1

        collection.update_one({"user_id": ctx.author.id},
        {"$set": {"last_fed": int(time()), "saturation": new_saturation, "mood": new_mood, "exp":
new_exp}})
        await ctx.send("Камень покормлен.")
        check_vars(ctx)
    else:
        await ctx.send("Камень еще не голоден.")

# Начать работу
@bot.group(invoke_without_command=True)
async def начать(ctx): ...
@начать.command()
async def работу(ctx):
    stone = collection.find_one({"user_id": ctx.author.id})
    if stone == None:
        await ctx.send("Друг, у тебя нет камня.")
        return

    if stone["activity"] != 0:
        await ctx.send("Камень уже чем то занят.")
    elif int(time()) - stone["ended_activity"] <= activity_cooldown:
        await ctx.send("Камню надо отдохнуть.")
    else:
        collection.update_one({"user_id": ctx.author.id},
        {"$set": {"started_activity": int(time()), "activity": 1}})
        await ctx.send("Камень отправлен на работу.")

# Завершить работу
```

```
@bot.group(invoke_without_command=True)
async def завершить(ctx): ...

@завершить.command()
async def работу(ctx):
    stone = collection.find_one({"user_id": ctx.author.id})
    if stone == None:
        await ctx.send("Друг, у тебя нет камня.")
        return

    if stone["activity"] != 1:
        await ctx.send("Камень не на работе.")
    elif int(time()) - stone["started_activity"] <= work_duration:
        await ctx.send("Камень еще не доработал.")
    else:
        if stone["mood"] <= 20:
            salary = randint(0, 30)
        elif stone["mood"] <= 50:
            salary = randint(10, 45)
        elif stone["mood"] <= 70:
            salary = randint(25, 65)
        else:
            salary = randint(25, 75)
        new_money = stone["money"] + salary
        new_mood = stone["mood"] - randint(7, 9)

        collection.update_one({"user_id": ctx.author.id},
        {"$set": {"ended_activity": int(time()), "activity": 0, "money": new_money, "mood":
new_mood}})
        await ctx.send(f"Камень вернулся с работы и принес {salary} монет.")
        check_vars(ctx)

# Покупки
@bot.group(invoke_without_command=True)
async def купить(ctx): ...
```

```
@купить.command()
async def конфеты(ctx, amount: int):
    stone = collection.find_one({"user_id": ctx.author.id})
    if stone == None:
        await ctx.send("Друг, у тебя нет камня.")
        return

    if stone["money"] - candy_price * amount < 0:
        await ctx.send("У тебя мало денег.")
    else:
        new_candies = stone["candies"] + amount
        new_money = stone["money"] - candy_price * amount
        collection.update_one({"user_id": ctx.author.id},
        {"$set": {"candies": new_candies, "money": new_money}})
        await ctx.send("Конфеты куплены.")

# Вылечить камень
@bot.group(invoke_without_command=True)
async def вылечить(ctx): ...
@вылечить.command()
async def камень(ctx):
    stone = collection.find_one({"user_id": ctx.author.id})
    if stone == None:
        await ctx.send("Друг, у тебя нет камня.")
        return

    if stone["candies"] < 2:
        await ctx.send("Нужно 2 конфеты.")
    elif stone["injured"] == 0:
        await ctx.send("Камень не поранен.")
    else:
        new_mood = stone["mood"] + randint(6, 9)
        collection.update_one({"user_id": ctx.author.id},
        {"$set": {"injured": 0, "mood": new_mood}})
        await ctx.send(f"Камень вылечен.")
        check_vars(ctx)
```

```
# Отправка
@bot.group(invoke_without_command=True)
async def отправить(ctx): ...

@отправить.command()
async def деньги(ctx, amount:int, user: discord.User):
    stone = collection.find_one({"user_id": ctx.author.id})
    if stone == None:
        await ctx.send("Друг, у тебя нет камня.")
        return
    user_stone = collection.find_one({"user_id": user.id})
    if stone == None:
        await ctx.send("Друг, у тебя нет камня.")
        return

    if stone["money"] < amount:
        await ctx.send("У тебя недостаточно монет.")
    else:
        new_sender_money = stone["money"] - amount
        new_user_money = user_stone["money"] + amount
        collection.update_one({"user_id": ctx.author.id},
        {"$set": {"money": new_sender_money}})
        collection.update_one({"user_id": user.id},
        {"$set": {"money": new_user_money}})
        await ctx.send(f"Деньги отправлены.")
        check_vars(ctx)

@отправить.command()
async def конфеты(ctx, amount:int, user: discord.User):
    stone = collection.find_one({"user_id": ctx.author.id})
    if stone == None:
        await ctx.send("Друг, у тебя нет камня.")
        return

    if stone["candies"] < amount:
```

```
        await ctx.send("У тебя недостаточно конфет.")
    else:
        new_sender_candies = stone["candies"] - amount
        new_user_candies = user_stone["candies"] + amount
        collection.update_one({"user_id": ctx.author.id},
        {"$set": {"candies": new_sender_candies}})
        collection.update_one({"user_id": user.id},
        {"$set": {"candies": new_user_candies}})
        await ctx.send(f"Конфеты отправлены.")
        check_vars(ctx)

# Запуск бота
bot.run(bot_token)
```