**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»**

**Журнал практики**

Студента *Бондаренко Владимира Дмитриевича*

##### Факультет Ступинский филиал МАИ

###### Кафедра *«Моделирование систем и информационные технологии»*

##### Учебная группа ТСО-105Б-22

Направление подготовки (специальность) *09.03.01*

*(шифр)*

*«Информатика и вычислительная техника»*

*(название направления, специальности)*

Вид практики *учебная*

*(учебной, производственной, преддипломной или другой вид практики)*

Руководитель практики от МАИ

*Мамонов Игорь Михайлович*  \_\_\_\_\_\_\_\_\_\_\_\_\_

*(фамилия, имя, отчество) (подпись)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/*Бондаренко В.Д.*/ “12” июля 2023г.

*(подпись студента)* *(дата)*

Москва 2023

**1.Место и сроки проведения практики**

*Сроки проведения практики:*

*-дата начала практики 29.06.2023 г.*

*-дата окончания практики 12.07.2023 г.*

*Наименование предприятия Ступинский филиал МАИ*

*Название структурного подразделения (отдел, лаборатория) кафедра «Моделирование систем и информационные технологии»*

1. **Инструктаж по технике безопасности**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Беспалов В.В./ “29” июня 2023 г.

*(подпись проводившего)* *(дата проведения)*

1. **Индивидуальное задание студенту**

Создание бота-собеседника для соц сети Telegram

**4.План выполнения индивидуального задания**

1. Изучение библиотек для создания чат-ботов(Конкретно TeleBot, Aiorgam).

2. Прописать основной функционал для проекта

3. Подготовить реферат.

*Руководитель практики от МАИ*: */Мамонов И.М./*

*Руководитель от предприятия*:  */ /*

*/Бондаренко В.Д./* “29” июня 2023 г.

*(подпись студента)* *(дата)*

**5.Отзыв руководителя практики от предприятия**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Руководитель от предприятия*: / /

*(подпись) (фамилия, имя, отчество)*

«\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

М.П. (печать)

**6.Отчет студента о практике**

Целью учебной практики являлось изучение API-библиотек для создания чат-бота, а так же последующее его сотворение.

В целях было сделать простого собеседника, который может реагировать на вариативные сообщения, типы файлов, а так же его собственный функционал(по типу рандомизации выбора фотографий и выбора случайного числа из изначально представленного списка).

Запрограмировать бота у меня получилось с помощью языка python в среде разработке PyCharm.

Python — [высокоуровневый язык программирования](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения с [динамической](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [строгой](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%BE%D0%B3%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости [кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью [объектно-ориентированным](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) в том плане, что всё является [объектами](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Необычной особенностью языка является выделение [блоков кода](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D0%B0) пробельными отступами. [Синтаксис](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как [интерпретируемый](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) и используется в том числе для написания [скриптов](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82).

**PyCharm** — это кроссплатформенная [интегрированная среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) для языка программирования python, разработанная компанией [JetBrains](https://ru.wikipedia.org/wiki/JetBrains" \o "JetBrains) на основе [IntelliJ IDEA](https://ru.wikipedia.org/wiki/IntelliJ_IDEA" \o "IntelliJ IDEA). Предоставляет пользователю комплекс средств для написания кода и визуальный [отладчик](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA). PyCharm предназначена для Python, но в ней есть поддержка и других языков. Например, Python часто используется в веб-разработке, поэтому IDE также поддерживает JavaScript для браузера и SQL для баз данных. Кроме JavaScript, поддерживаются основанные на нем TypeScript и CoffeeScript, популярные JS-фреймворки, а также языки HTML и CSS для верстки.

Установка библиотек

Для установки мы воспользуемся уже установленными средой PyCharm и python

Pip install - позволяет взять определённый ресурс из сети в внедрить его в Вашу среду разработки.

Для установки Telebot’а(или aiogram) аргументом для этой программы будет служить название конкретных библиотек

Напрмер:

Pip install telebot прописываем в терминал

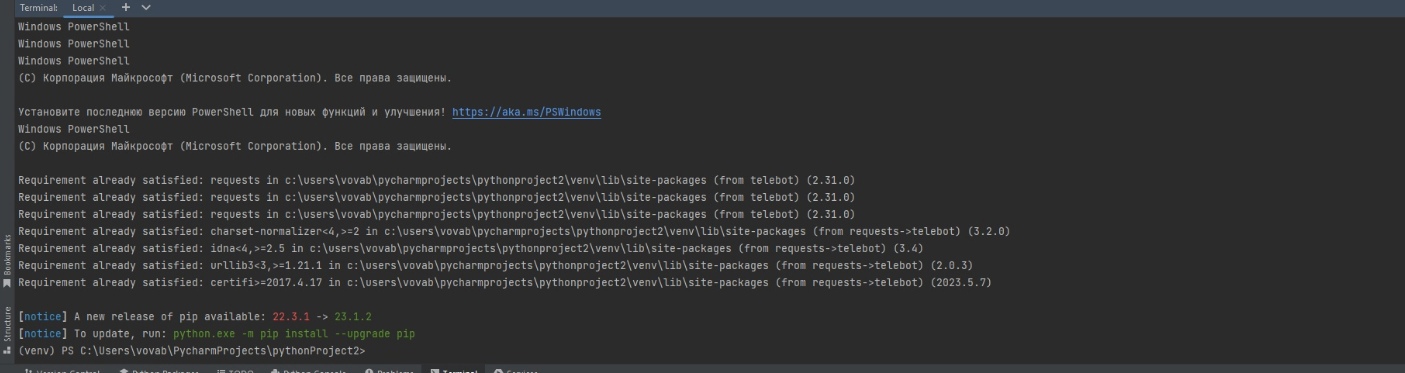


Рис 1. Установка библиотеки

В данном случае библиотека просто обновилась, ибо до этого она уже была установлена. В целом, принцип тут один и тот же.

Получение токена и регистрация бота.

Для получения токена используем BotFather - это система, позволяющая нам зарегистрировать собственную

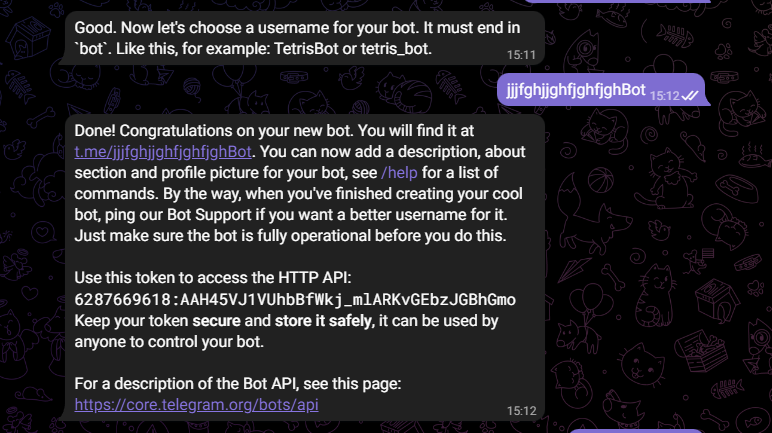


Рис 3. Получение токена

Написание программы.

На данном этапе мы приступаем к самому написанию. Во-первых(это важно) нужно произвести импорт библиотеки, но, в целом, это не так трудно.

import random

import telebot

from telebot import types

bot = telebot.TeleBot('6287669618:AAH45VJ1VUhbBfWkj\_mlARKvGEbzJGBhGmo')

Далее стоит прописать

bot = telebot.TeleBot('6287669618:AAH45VJ1VUhbBfWkj\_mlARKvGEbzJGBhGmo')

Данная конструкция позволяет нам отсылаться на существующий токен, а точнее зарегистрированного бота, с которым в дальнейшем будут производиться те или иные манипуляции. Данная конструкция может иметь различные вариации, все зависит от выбора библиотеки и изучения ее же синтаксиса.

Что касается изучения основ библиотеки. Использовался <https://docs.python-telegram-bot.org/en/stable/>, сайт, где представлена официальная документация по изучению основных функций.

@bot.message\_handler(commands=['start'])

def start(message):

    bot.send\_message(message.chat.id, "", parse\_mode='html')

@bot.message\_handler(content\_types=['photo'])

def u\_photo(message):

    bot.send\_message(message.chat.id, 'Какая классная картинка!')

@bot.message\_handler(content\_types=['video'])

def u\_video(message):

    bot.send\_message(message.chat.id, 'Какое классное видео!')

**@bot.message\_handler** по своей сути явяется декоратором для функции, в качестве аргумента для которого даётся фильтр отбора входящих сообщений. Тоесть, каждый раз когда боту приходит сообщение, он проверяет проходит оно по фильтрам или нет. Если подходит - запускает декорируемую функцию. В данном случае здесь представлены несколько типов аргументов, commands

Представляет собой ожидание команды, по умолчанию они прописываются через / в непосредственном чате



Рис 2. Пример команды.

Или же, если в аргументе стоит тип файла content\_types=['photo']

content\_types=['video']

Бот будет выполнять определённые действия, при отправке ему того или иного файла(в данном случае фотографии и видеозаписи).

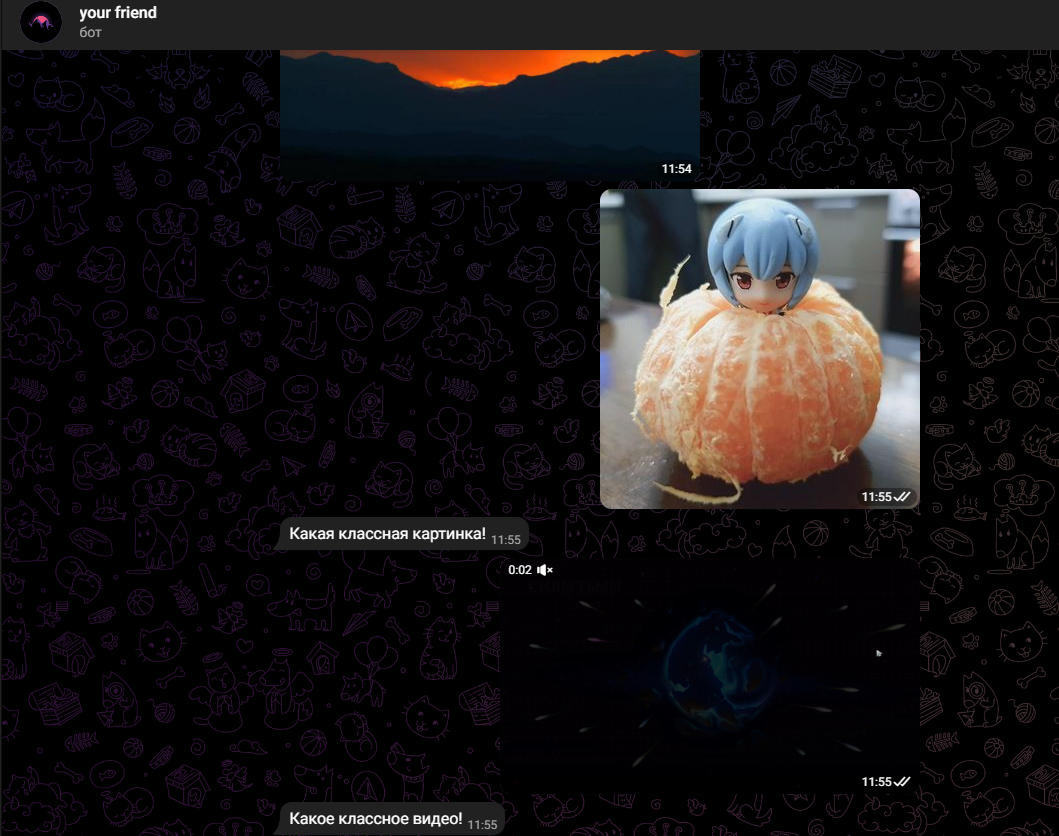


Рис 3. Реакции на медиа-файлы

Bot.send\_message является некой «обратной связью», благодаря которой бот способен выдавать ответную реакцию.

@bot.message\_handler()

def get\_user\_text(message):

    if message.text == "hello" or message.text == "Hello" or message.text == "привет" or message.text == "Привет" or message.text == "здравствуй" or message.text == "Здравствуй":

        bot.send\_message(message.chat.id, "приветики", parse\_mode='html')

    elif message.text == "как дела":

        bot.send\_message(message.chat.id, "не знаю, я, в принципе, всего лишь робот, который всегда будет чувствовать себя, допустим, нормально. Надеюсь, тебя устроит мой ответ", parse\_mode='html')

    elif message.text == "чем занимаешься?" or message.text == "что делаешь?":

        bot.send\_message(message.chat.id, "радуюсь каждой минуте своего сущестования, а так же общаюсь с тобой", parse\_mode='html')

    elif message.text == "скинь котика":

        photo =  open('cat.jpeg', 'rb')

        bot.send\_photo(message.chat.id, photo)

    elif message.text == "что ты умеешь" or message.text == "что ты еще умеешь":

        markup = types.InlineKeyboardMarkup()

        markup.add(types.InlineKeyboardButton("выдать рандомное фото", callback\_data='img\_path'))

        markup.add(types.InlineKeyboardButton("сгенерировать случайное число", callback\_data='test'))

        bot.send\_message(message.chat.id, 'Вот что я умею', reply\_markup=markup)

    else:

        bot.send\_message(message.chat.id, "Не понимаю тебя, можешь написать ЧТО ТЫ УМЕЕШЬ для выбора операций", parse\_mode='html')

Ответ от бота будет получен только в том случае, если фраза будет соответстовать легитности базы данных, в противном случае(else) будет выдана подсказка «Не понимаю тебя».

Заметим, что использовалась конструкция callback\_data

Позволяющая нам отослаться на определнную функцию, в моем случае - это рандомизация

@bot.callback\_query\_handler(func=lambda c: c.data == "test")

def rand(c):

    a = random.randint(0,100000000)

    bot.edit\_message\_text(text=str(a),chat\_id=c.from\_user.id,message\_id=c.message.id)

@bot.callback\_query\_handler(func=lambda d: d.data =='img\_path')

def choose(d):

    img\_list = ['cats/cat.jpeg', 'cats/cat2.jpg', 'cats/cat3.jpg','cats/moon.jpg','cats/sun.jpg']

    img\_chsoe = random.choice(img\_list)

    bot.send\_photo(photo=open(img\_chsoe, 'rb'), chat\_id=d.from\_user.id)

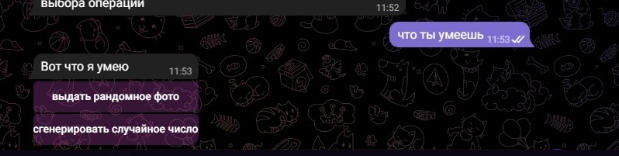




Рис 8. Рандомайзер числа и фотографий.

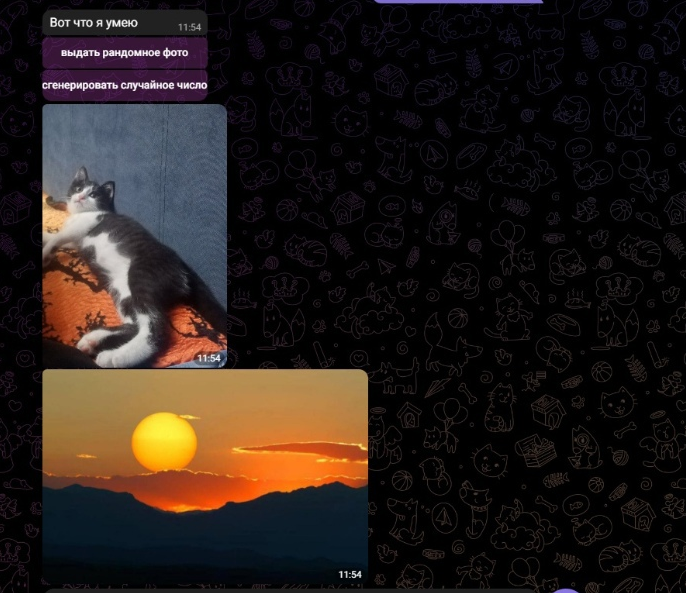


Рис 9. Рандомазйер на примере(фотографии)

По итогу у меня получилось создать простого бота собеседника, отвечающего на те сообщения, которые присутствуют у него в базе данных, либо же на что-то похожее из них.

1. Изучена библиотека Telebot.
2. Практическое занятие с тестированием некоторых функций, получением токена.
3. Конечный результат.