## Создание чат-ботов для социальной сети Telegram

Все многообразие чат-ботов можно разделить на 2 большие категории:

* работающие по заранее известным командам на основе ограниченного списка ключевых слов. Их гораздо проще создавать, однако практическое применение таких чат-ботов ограничено ключевыми словами, при отсутствии которых в запросе бот не сможет помочь пользователю.
* самообучающиеся на базе алгоритмов Machine Learning и методов понимания естественного языка (NLU, Natural Language Understanding). Создавать такие боты намного сложнее, однако на практике они оказываются намного эффективнее за счет накопления знаний на основе предыдущих взаимодействий с пользователем.

## Как работают чат-боты

Типовой цикл работы любого чат-бота можно представить цепочкой следующих действий:

1. получение запроса от клиента;
2. разбор запроса – понимание высказывания и определение намерений клиента в контексте его бизнес-кейса;
3. выполнение действий согласно заранее определенному сценарию (скрипту) по обработке клиентского кейса;
4. генерация ответа на естественном языке;
5. сохранение запроса, контекста и параметров диалога для обработки последующих обращений;
6. отправка ответа клиенту.

Наиболее сложным этапом работы является разбор клиентского запроса. Самообучающиеся чат-боты на базе Machine Learning используют для этого методы NLU и NLP. Например, для текстовых чат-ботов процесс разбора включает следующие этапы:

* предварительная обработка текста – токенизация (разбиение на слова), исправление опечаток, лемматизация и стемминг (определение нормальной формы слов и частей речи), отбрасывание стоп-слов (артикли, междометия, союзы и пр.), расширение запроса с помощью - словарей синонимов, дополнение информацией о значимости отдельных слов, расширение запроса деревом синтаксического разбора и результатами разрешения местоимений, а также определение именованных сущностей.
* [классификация](https://www.bigdataschool.ru/wiki/%d0%ba%d0%bb%d0%b0%d1%81%d1%81%d0%b8%d1%84%d0%b8%d0%ba%d0%b0%d1%86%d0%b8%d1%8f) запроса на основе примеров фраз и ML-алгоритмов или формальных правил (шаблонов), ранжирование гипотез классификации в соответствии с текущим контекстом беседы;
* извлечение параметров запроса из фразы пользователя.

Как правило, чат-боты имеют следующую архитектуру:

* серверная часть (backend), где собственно и выполняются работы по распознаванию запроса и формированию ответа;
* клиентская часть (frontend), обычно в виде мессенджера ([Telegram](https://telegram.org/" \t "_blank) и пр.).

Боты — это неболь­шие скрип­ты, которые могут вза­имо­дей­ство­вать с API, что­бы получать сооб­щения от поль­зовате­ля и отправ­лять информа­цию в раз­ные чаты и каналы.

## Лабораторное задание

## Создание бота

Что­бы соз­дать бота, нуж­но дать ему наз­вание, адрес и получить токен — стро­ку, которая будет однознач­но иден­тифици­ровать бота для сер­веров Telegram. Зай­дем в Telegram под сво­им акка­унтом и откро­ем «отца всех ботов», [BotFather](https://t.me/BotFather).

Жмем кноп­ку «Запус­тить» (или отпра­вим /start), в ответ BotFather приш­лет нам спи­сок дос­тупных команд:

* /newbot — соз­дать нового бота;
* /mybots — редак­тировать ваших ботов;
* /setname — сме­нить имя бота;
* /setdescription — изме­нить опи­сание бота;
* /setabouttext — изме­нить информа­цию о боте;
* /setuserpic — изме­нить фото ава­тар­ки бота;
* /setcommands — изме­нить спи­сок команд бота;
* /deletebot — уда­лить бота.

От­пра­вим BotFather коман­ду /newbot, что­бы соз­дать нового бота. В ответ он поп­росит ввес­ти имя будуще­го бота, его мож­но писать на рус­ском. Пос­ле вво­да име­ни нуж­но будет отпра­вить адрес бота, при­чем он дол­жен закан­чивать­ся на сло­во bot. Нап­ример, xakepbot или xakep\_bot. Если адрес будет уже кем‑то занят, BotFather нач­нет изви­нять­ся и про­сить при­думать что‑нибудь дру­гое.

Ког­да мы най­дем сво­бод­ный адрес для бота, в ответ получим сооб­щение, в котором пос­ле фра­зы Use this token to access the HTTP API будет написа­на стро­ка из букв и цифр — это и есть необ­ходимый нам токен. Сох­раним ее где‑нибудь на сво­ем компь­юте­ре, что­бы потом исполь­зовать в скрип­те бота.

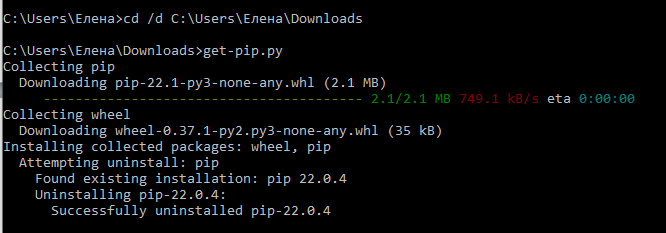
## Подготовка к работе над ботом

1. Для выполнения лабораторной работы необходимо скачать Python, можно [с официального сайта](https://www.python.org/). Подойдет Python не ниже версии 3.7. Если питон уже установлен, то узнать версию можно с помощью командной строки. Для этого нажимаем клавиши Windows + R на клавиатуре, вводим **cmd** и нажимаем «ОК». В командной строке ввести: python --version.



1. Чтобы установить готовый модуль для работы с Telegram API, необходимо установить менеджер пакетов pip, если он не установлен.

* Скачайте [установочный скрипт get-pip.py](https://bootstrap.pypa.io/get-pip.py).
* Откройте командную строку и перейдите к каталогу с файлом get-pip.py. Чтобы открыть каталог с нужным файлом в командной строке пишем cd /d «полный путь каталога»
* Запустите следующую команду: get-pip.py



1. Для вза­имо­дей­ствия с Telegram API есть нес­коль­ко готовых модулей. Самый прос­той из них — Telebot. Необходимо установить модуль pyTelegramBotAPI, наберите в командной строке: pip install pytelegrambotapi

Для некоторых ботов необходимо будет установить дополнительно другие пакеты. Но для начала будет достаточно этого. Далее необходимо открыть любой редактор кода, где будем писать самого бота.

## Знакомимся с Python

Ниже мы рассмотрим компоненты, которые необходимы любому боту вне зависимости от его функционала.

Подключаем модуль telebot, который мы установили с помощью pip. Делаем это так:

import telebot

При создании бота, мы получили Token бота. Это ключ, с помощью которого мы сможем управлять нашим ботом через Python.

Присваиваем переменной TOKEN наш ключ-строку:

TOKEN = 'Здесь впишите токен, полученный от @botfather '

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

bot = telebot.TeleBot(TOKEN)

Либо можно присвоить токен одной строкой

bot = telebot.TeleBot('Здесь впишите токен, полученный от @botfather')

Вы можете просто передать свой токен бота здесь, но если вы планируете загрузить свой код на Github или поделиться с кем-либо, не рекомендуется публично размещать свой токен бота там. Можно сохранть его в  файле **.env** . Для этого сначала установить его библиотеку с именем [python-dotenv](https://laptrinhx.com/link/?l=https%3A%2F%2Fpypi.org%2Fproject%2Fpython-dotenv%2F" \t "_blank).

pip install python-dotenv

Затем создать  файл *.env* в том же каталоге, что и ваш файл *Bot.py* , и в нем определить свой токен бота.

BOT\_TOKEN = "your\_bot\_token\_here"

И теперь в файле Bot.py надо импортировать библиотеку Dot-Env. Затем мы используем функцию *getenv()* , присутствующую в os, передав имя, использованное нами выше, в  файле **.env** в качестве параметра. После доступа к токену бота из файла .env мы передадим его Telebot, чтобы он понял, какие команды Telegram-бота он должен слушать.

import telebot  
import os  
from dotenv import load\_dotenv

import telebot  
import os  
from dotenv import load\_dotenv

Наша библиотека pyTelegramBotAPI (модуль для Python) поддерживает декораторы (или хэндлеры), с помощью которых можно научить бота отвечать на различные сообщения. Создадим декоратор, который будет отвечать на команду /start (эта команда соответствует нажатию кнопки Start или Запустить, в профиле бота в Telegram); внутри обработчика создадим функцию, в которой будем спрашивать имя пользователя.

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

def start(message):

Вы можете определить функции для обеспечения требуемой функциональности. Вот простые правила для определения функции в Python.

* Функциональные блоки начинаются с ключевого слова **def** , за которым следует имя функции и круглые скобки '( )'.
* Любые входные параметры или аргументы должны быть заключены в эти круглые скобки. Вы также можете определить параметры внутри этих скобок.
* Первый оператор функции может быть необязательным оператором — строкой документации функции или строкой документации.
* Блок кода внутри каждой функции начинается с двоеточия ':' и имеет отступ.
* Оператор return [выражение] выходит из функции, при необходимости возвращая выражение вызывающей стороне. Оператор return без аргументов аналогичен return None.

Конструкция if/elif/else позволяет делать ответвления в ходе программы. Программа уходит в ветку при выполнении определенного условия.

В этой конструкции только if является обязательным, elif и else опциональны:

* Проверка if всегда идет первой.
* После оператора if должно быть какое-то условие: если это условие выполняется (возвращает True), то действия в блоке if выполняются.
* С помощью elif можно сделать несколько разветвлений, то есть, проверять входящие данные на разные условия.
* Блок elif это тот же if, но только следующая проверка. Грубо говоря, это «а если …»
* Блоков elif может быть много.
* Блок else выполняется в том случае, если ни одно из условий if или elif не было истинным.

Пример конструкции:

**if** a == 10:

print('a равно 10')

**elif** a < 10:

print('a меньше 10')

**else**:

print('a больше 10')

Метод append() в Python добавляет элемент в конец списка.

Пример конструкции:

inventories[user].append("key") #добавление в список inventories ключа

Оператор in  проверяет наличие элемента в последовательности. Выражение x in s принимает значение True, если x является членом s, и False в противном случае.

Пример конструкции:

if "key" in inventories[user]: # проверка на наличие ключа в инвентаре.

Метод randint()возвращает целое число выбранного элемента из указанного диапазона.

Пример конструкции:

chance = randint(0, 100) #переменной chance присваивается рандомное число из диапазона от 0 до 100

Сущности &lt; и  &gt; используются для ‘<’ и  ‘>’.

Пример конструкции:

if chance &gt; 30: # если chance > 30

Для того, чтобы отправить сообщение в какой-либо чат нам необходимо указать несколько параметров в функцию send\_message. Первый параметр — это chat\_id, собственно сам id чата. Второй — text, текст сообщения. Вместо того, чтобы писать message.chat.id, мы можем написать свои данные! Вот так можно прислать сообщение самому себе, если указать в параметрах свой id:

bot.send\_message(<ваш id>, 'Текст сообщения')

Класс [CallbackQueryHandler(callback)](https://docs-python.ru/packages/biblioteka-python-telegram-bot-python/menju-klaviatury/" \o "Меню из кнопок, модуль python-telegram-bot в Python.) предназначен для обработки запросов обратного вызова Telegram (нажатие кнопок встроенной клавиатуры):

@bot.callback\_query\_handler(func=lambda call: True)

Чтобы наш бот мог принимать ответы от пользователя, он будет у сервера Telegram постоянно спрашивать "Мне кто-нибудь написал?" Наш бот получил переданное сообщение. Для этого в конце кода допишем:

bot.polling(none\_stop=True)

## Запуск бота

После того как код бота написан, необходимо сохранить его с расширением py. Чтобы запустить бота, в командной строке переходим к каталогу, в котором расположен файл с нашим ботом и вводим его название. Если не было выведено никаких сообщений об ошибке, то бот запустился и готов к работе.



Переходим в Telegram и даем боту команду /start .

## ПримерTelegram бота

Чат-бот «Квест»

Данный бот предлагает пользователю пройти текстовый квест.

from telebot import TeleBot, types #импорт библиотек

from random import randint

pictures = {#массив для хранения ссылок на изображения локаций

0: "https://cdn.ryfma.com/images/ozJtmxqwo9PBpAMWC/posts/post\_full\_1601737968052\_podzemele\_1-jpg.jpg",

1: "https://kazpravda.kz/media/uploads/publication2/133/40/133402-image.jp",

2: " https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRJlgw-e-LZNON-mpGTmebVtKH4qmIASMWXRQ&usqp=CAU"

}

states = {}#локация,в которой находится пользователь

inventories = {}#инвентарь игрока

token = '5386164634:AAGfJYFlPusL-\_1Or4Wv9czEOCSWRsm\*\*\*\*' # токен бота, полученный от @botfather

bot = TeleBot(token)

@bot.message\_handler(commands=["start"]) #запуск работы бота после команды /start

def start\_game(message):

user = message.chat.id #переменная для хранения id чата, чтобы отправлять ему сообщения

states[user] = 0 #определяем, что пользователь находится на начальной локации 0

inventories[user] = [] #у пользователя пока нет никакого инвентаря

bot.send\_message(user, "Добро пожаловать в игру!") #отправка сообщения пользователю

process\_state(user, states[user], inventories[user]) #вызов функции process\_state

@bot.callback\_query\_handler(func=lambda call: True) #обработка запросов обратного вызова Telegram (нажатие кнопок встроенной клавиатуры)

def user\_answer(call):

user = call.message.chat.id

process\_answer(user, call.data)#вызов функции process\_answer

def process\_state(user, state, inventory):

kb = types.InlineKeyboardMarkup() #создание кнопок

bot.send\_photo(user, pictures[state]) #отправка изображения

if state == 0:#проверка на номер локации

kb.add(types.InlineKeyboardButton(text="пойти направо", callback\_data="1")) #добавление кнопки

kb.add(types.InlineKeyboardButton(text="пойти налево", callback\_data="2"))

bot.send\_message(user, "Вы в оказались в темном подземелье, перед вами два прохода.", reply\_markup=kb)

#

#

# Продолжите описание ваших локаций

#

#

if state == """вставьте номер вашей последней локации""":

bot.send\_message(user, "Вы выиграли.")

def process\_answer(user, answer):

if states[user] == 0: #Проверка на номер локации

if answer == "1": #Проверка ответа который дал пользователь

states[user] = 1 #Присвоение нового номер локации и переход на новую локацию

else:

bot.send\_message(user, "Перед вами закрытая дверь. Вы пробуете открыть ее ключом, и дверь поддается. Кажется, это выход.")

states[user] = 2

elif states[user] == 1:#Проверка на номер локации

#

#

# Продолжите описание реакций бота на ответ пользователя

#

#

#

process\_state(user, states[user], inventories[user]) #вызов функции process\_state

bot.polling(none\_stop=True) )#проверка на наличие ответа от пользователя

## Задание

## Добавьте к квесту не менее 7 локаций.

## Реализовать успех совершения действия с использованием функции рандомной генерации числа (например, вероятность прохождения препятствия).

## Реализовать использование списка inventories, путем получения в ходе прохождения квеста инвентаря, необходимого для успешного прохождения другой локации.

## Добавить подходящие для вашего квеста изображения для каждой локации.

## Предложить пользователю в ходе квеста решить какую-либо логическую задачу, пользователь не сможет продолжить выполнение квеста пока не ответит правильно.

## Тематика квеста может быть изменена по вашему желанию, но вышеуказанные требования должны быть выполнены. Перейти к разделу “Загрузка на бесплатный сервер”, загрузить бот перед демонстрацией работы.

## Изучить и продемонстрировать работу следующих ботов:

**ЭХО-БОТ**

Реали­зация эхо‑бота. Он будет получать от поль­зовате­ля тек­сто­вое сооб­щение и воз­вра­щать его.

import telebot

# Создаем экземпляр бота

bot = telebot.TeleBot('Здесь впишите токен, полученный от @botfather')

# Функция, обрабатывающая команду /start

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

def start(m, res=False):

bot.send\_message(m.chat.id, 'Я на связи. Напиши мне что-нибудь )')

# Получение сообщений от юзера

@bot.message\_handler(content\_types=["text"])

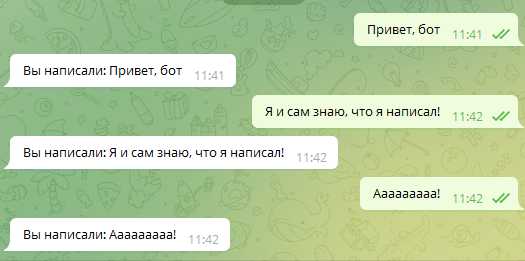
def handle\_text(message):

bot.send\_message(message.chat.id, 'Вы написали: ' + message.text)

# Запускаем бота

bot.polling(none\_stop=True, interval=0)

За­пус­кайте скрипт и ищите в поис­ке Telegram сво­его бота по адре­су, который вы при­думали ранее. Запус­каем бота кноп­кой «Запус­тить» (Start) или коман­дой /start и можем убе­дить­ся в том, что он работа­ет и воз­вра­щает сооб­щения.

Пример работы эхо-бота:

**WIKIPEDIA-БОТ**

Теперь научим бота не прос­то отсы­лать сооб­щения обратно, а, например, по вве­ден­ному сло­ву давать статью на Википе­дии. Здесь поможет модуль **Wikipedia**:

pip install wikipedia

Го­товим код.

import telebot, wikipedia, re

# Создаем экземпляр бота

bot = telebot.TeleBot('Здесь впишите токен, полученный от @botfather')

# Устанавливаем русский язык в Wikipedia

wikipedia.set\_lang("ru")

# Чистим текст статьи в Wikipedia и ограничиваем его тысячей символов

def getwiki(s):

try:

ny = wikipedia.page(s)

# Получаем первую тысячу символов

wikitext=ny.content[:1000]

# Разделяем по точкам

wikimas=wikitext.split('.')

# Отбрасываем всЕ после последней точки

wikimas = wikimas[:-1]

# Создаем пустую переменную для текста

wikitext2 = ''

# Проходимся по строкам, где нет знаков «равно» (то есть все, кроме заголовков)

for x in wikimas:

if not('==' in x):

# Если в строке осталось больше трех символов, добавляем ее к нашей переменной и возвращаем утерянные при разделении строк точки на место

if(len((x.strip()))>3):

wikitext2=wikitext2+x+'.'

else:

break

# Теперь при помощи регулярных выражений убираем разметку

wikitext2=re.sub('\([^()]\*\)', '', wikitext2)

wikitext2=re.sub('\([^()]\*\)', '', wikitext2)

wikitext2=re.sub('\{[^\{\}]\*\}', '', wikitext2)

# Возвращаем текстовую строку

return wikitext2

# Обрабатываем исключение, которое мог вернуть модуль wikipedia при запросе

except Exception as e:

return 'В энциклопедии нет информации об этом'

# Функция, обрабатывающая команду /start

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

def start(m, res=False):

bot.send\_message(m.chat.id, 'Отправьте мне любое слово, и я найду его значение на Wikipedia')

# Получение сообщений от юзера

@bot.message\_handler(content\_types=["text"])

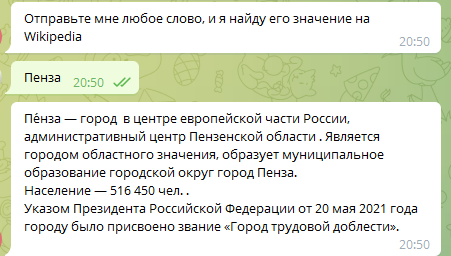
def handle\_text(message):

bot.send\_message(message.chat.id, getwiki(message.text))

# Запускаем бота

bot.polling(none\_stop=True, interval=0)

Пример работы Wikipedia-бота



**БОТ С ДВУМЯ ВИРТУАЛЬНЫМИ КНОПКАМИ**

Во мно­гих Telegram-ботах для выбора каких‑то дей­ствий исполь­зуют­ся так называ­емые вир­туаль­ные кноп­ки.

Создайте и заполните два фай­ла facts.txt и thinks.txt, которые содер­жат спи­сок инте­рес­ных фак­тов и поговор­ки. На каж­дой стро­ке фай­лов должно находит­ся по одно­му фак­ту или поговор­ке.

Сде­лаем бота, в котором будут две кноп­ки: «Фак­ты» и «Поговор­ки». Если нажать любую, бот отпра­вит поль­зовате­лю соот­ветс­тву­ющее сооб­щение.

Ес­ли исполь­зовать для это­го бота тот же токен, что и для пре­дыду­щего, что­бы уви­деть кноп­ки, необходимо переза­пус­тить бота коман­дой /start.

import telebot

import random

from telebot import types

# Загружаем список интересных фактов

f = open('data/facts.txt', 'r', encoding='UTF-8')

facts = f.read().split('\n')

f.close()

# Загружаем список поговорок

f = open('data/thinks.txt', 'r', encoding='UTF-8')

thinks = f.read().split('\n')

f.close()

# Создаем бота

bot = telebot.TeleBot('Здесь твой токен, полученный от @botfather')

# Команда start

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

def start(m, res=False):

# Добавляем две кнопки

markup=types.ReplyKeyboardMarkup(resize\_keyboard=True)

item1=types.KeyboardButton("Факт")

item2=types.KeyboardButton("Поговорка")

markup.add(item1)

markup.add(item2)

bot.send\_message(m.chat.id, 'Нажмите: \nФакт для получения интересного факта\nПоговорка — для получения мудрой цитаты ', reply\_markup=markup)

# Получение сообщений от юзера

@bot.message\_handler(content\_types=["text"])

def handle\_text(message):

# Если юзер прислал 1, выдаем ему случайный факт

if message.text.strip() == 'Факт' :

answer = random.choice(facts)

# Если юзер прислал 2, выдаем умную мысль

elif message.text.strip() == 'Поговорка':

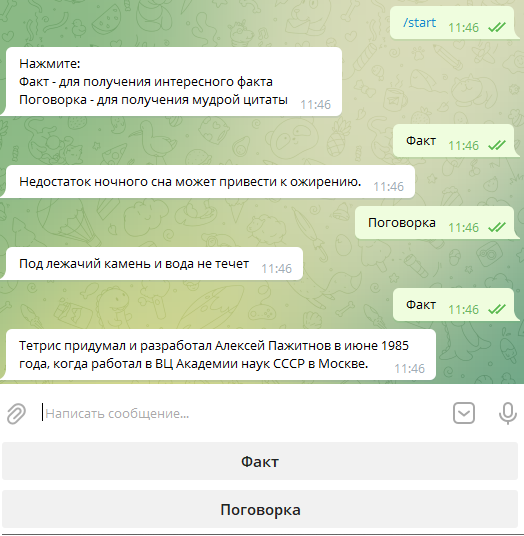
answer = random.choice(thinks)

# Отсылаем юзеру сообщение в его чат

bot.send\_message(message.chat.id, answer)

# Запускаем бота

bot.polling(none\_stop=True, interval=0)

Пример работы бота, знающего факты и поговорки

**БОТ, ВЕДУЩИЙ TELEGRAM-КАНАЛ С АНЕКДОТАМИ**

Пре­дыду­щие боты посыла­ли юзе­ру сооб­щения тог­да, ког­да получа­ли от него коман­ды или фра­зы. Но что, если нам нужен бот, который будет пери­оди­чес­ки и в авто­мати­чес­ком режиме пос­тить что‑то в наш канал?

Сде­лаем бота, который получа­ет спи­сок анек­дотов из фай­ла и каж­дый час пос­тит в канал один из этих анек­дотов. Для это­го нам нуж­но соз­дать свой канал в Telegram, добавить в под­писчи­ки канала нашего бота и наз­начить его адми­нис­тра­тором канала с пра­вом пуб­ликовать сооб­щения.

Создайте файл с анек­дотами, он дол­жен лежать в пап­ке data рядом со скрип­том бота.

import telebot

import time

# Токен, который выдает @botfather

bot = telebot.TeleBot('Здесь токен, полученный от @botfather')

# Адрес телеграм-канала, начинается с @

CHANNEL\_NAME = '@адрес\_вашего\_канала'

# Загружаем список шуток

f = open('data/fun.txt', 'r', encoding='UTF-8')

jokes = f.read().split('\n')

f.close()

# Пока не закончатся шутки, посылаем их в канал

for joke in jokes:

bot.send\_message(CHANNEL\_NAME, joke)

# Делаем паузу в один час

time.sleep(3600)

bot.send\_message(CHANNEL\_NAME, "Анекдоты закончились :-(")

**ЧАТ-БОТ «МАША»**

Сде­лаем прос­тей­шего чат‑бота, который будет разговаривать с поль­зовате­лем. Для это­го мы под­готовим файл boltun.txt, содер­жащий стро­ки с воп­росами (в начале таких строк пос­тавим мет­ку u:) и отве­тами на них в сле­дующей стро­ке.

u: как зовут

Маша меня зовут!

u: сколько тебе лет

Мне 18!

Файл boltun.txt помес­тим в пап­ку data рядом со скрип­том бота. Для поис­ка похожих воп­росов исполь­зуем модуль **fuzzywuzzy**, который поз­воля­ет срав­нивать, нас­коль­ко похожи меж­ду собой две стро­ки. Естес­твен­но, спер­ва этот модуль нуж­но уста­новить:

pip install fuzzywuzzy

pip install python-Levenshtein

Ни­же при­веден исходный код бота. Пос­ле его запус­ка напишите боту «При­вет» и поп­робуйте с ним пооб­щать­ся. Естес­твен­но, это не искусс­твен­ный интеллект и набор его отве­тов огра­ничен фра­зами из фай­ла boltun.txt.

import telebot

import os

from fuzzywuzzy import fuzz

# Создаем бота, пишем свой токен

bot = telebot.TeleBot('Здесь токен, полученный от @botfather')

# Загружаем список фраз и ответов в массив

mas=[]

if os.path.exists('data/boltun.txt'):

f=open('data/boltun.txt', 'r', encoding='UTF-8')

for x in f:

if(len(x.strip()) > 2):

mas.append(x.strip().lower())

f.close()

# С помощью fuzzywuzzy вычисляем наиболее похожую фразу и выдаем в качестве ответа следующий элемент списка

def answer(text):

try:

text=text.lower().strip()

if os.path.exists('data/boltun.txt'):

a = 0

n = 0

nn = 0

for q in mas:

if('u: ' in q):

# С помощью fuzzywuzzy получаем, насколько похожи две строки

aa=(fuzz.token\_sort\_ratio(q.replace('u: ',''), text))

if(aa > a and aa!= a):

a = aa

nn = n

n = n + 1

s = mas[nn + 1]

return s

else:

return 'Ошибка'

except:

return 'Ошибка'

# Команда «Старт»

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

def start(m, res=False):

bot.send\_message(m.chat.id, 'Я на связи. Напиши мне Привет )')

# Получение сообщений от юзера

@bot.message\_handler(content\_types=["text"])

def handle\_text(message):

# Запись логов

f=open('data/' + str(message.chat.id) + '\_log.txt', 'a', encoding='UTF-8')

s=answer(message.text)

f.write('u: ' + message.text + '\n' + s +'\n')

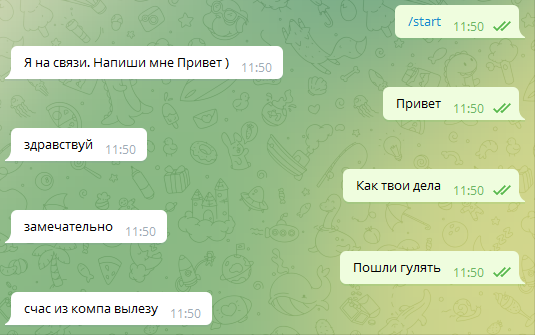
f.close()

# Отправка ответа

bot.send\_message(message.chat.id, s)

# Запускаем бота

bot.polling(none\_stop=True, interval=0)

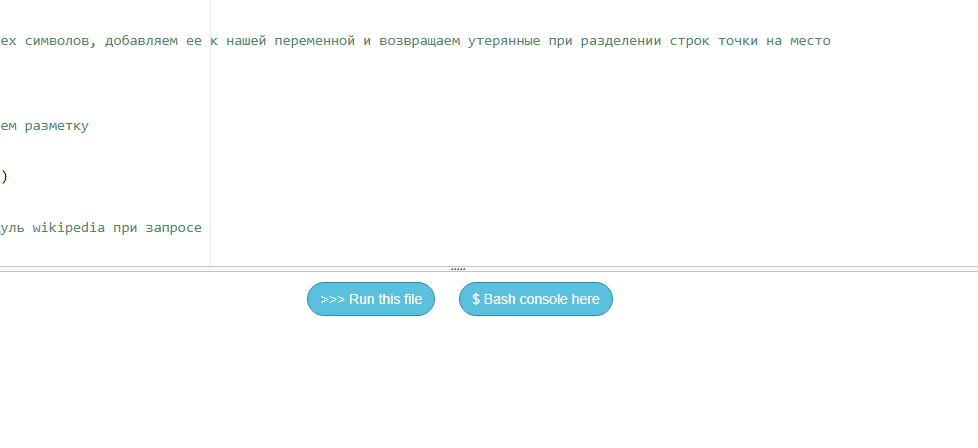
Пример работы чат-бота

Загрузка на бесплатный сервер

1. Зайти на сайт <https://www.pythonanywhere.com>
2. Пройти процесс регистрации (выбрать Create a Beginner account)
3. Открыть Consoles -> Bash
4. (проверить версию pip –V )
5. [Пакет python-telegram-bot](https://docs-python.ru/packages/biblioteka-python-telegram-bot-python/) предоставляет чистый интерфейс Python для [Telegram Bot API](https://core.telegram.org/bots/api). Он совместим с версиями Python 3.6+.

Установить библиотеку (пакеты установятся под вашего временного пользователя)

pip3 install python-telegram-bot --user

1. Перейти в Files->Upload a file->Открыть необходимый файл (напр. Bot.py) ->Run this file
   1. 
2. Необходимые модули устанавливать Consoles->bash.consoles