Практическое занятие к тренингу «Основы программирования на Python» SCRIPT-002FL

Цель практики: закрепить полученные теоретические знания

Постановка задачи: Мы будем разрабатывать бота для телеграмма. Бот будет поддерживать разговор, искать плохие слова и выполнять ваши задания

1. Настройка окружения и начальный код

- Установите телеграмм и создайте свой аккаунт, если у вас его еще нет
- Создайте собственного бота и получите его токен. Подробности по ссылке https://tlgrm.ru/docs/bots#botfather
- В вашей среде разработки приложений создайте файл main.py
- В первых строках файла импортируйте необходимые модули:

```
from telegram.ext import Updater, CommandHandler,MessageHandler,Filters,ConversationHandler
from telegram import ReplyKeyboardMarkup, ReplyKeyboardRemove
```

- Определите функцию start(), которая будет запускаться по команде /start, отправленной пользователю ботом. В этой функции мы получим данные пользователя, отобразим их и поприветствуем пользователя, подключившего наш бот:

```
def start(update,context):
    user = update.message.from_user
    send = f"{user.username} started you bot.\n First name {user.first_name}\n ID:{user.id}"
    context.bot.send_message(chat_id = user.id,text = send)
    update.message.reply_text('Hi')
```

- Определите сразу после функции переменную токен, куда в виде строки сохраните ваш токен, полученный от @BotFather телеграма

```
TOKEN =  “YOU TOKEN HERE”
```

- Создайте подключение бота к телеграму и получите ссылку на встроенного диспетчера, который отвечает за маршрутизацию получаемых ботом сообщений:

```
updater = Updater(TOKEN,use_context=True)
dp = updater.dispatcher
```

- Добавьте как обработчик команды вашу функцию start():

```
dp.add_handler(CommandHandler("start",start))
```

- Вызовите цикл обработки вашего бота:

```
updater.start_polling()
updater.idle()
```

- Запустите программу. Обратите внимание, что бот работает пока программа не завершена. Функция idle() не возвращает управление! Завершить работу программы можно будет по нажатию Ctrl-Break
- Найдите вашего бота в телеграмме, запустите его и введите команду /start
- Проанализируйте полученный результат

Мы написали простейшего телеграм-бота!

2. Фильтр плохих слов

Пользователи бывают разные по характеру. Не все пишут вежливо. Давайте напишем задачу, которая будет фильтровать плохие слова и возвращать пользователю текст, заменяя плохие слова на «***».

- Определите функцию bad_words_filter(), которая принимает исходную строку и возвращает строку, где плохие слова заменены на «***»
- Определите функцию words_filter_handler(), которая будет вызывать фильтр плохих слов, а измененный текст возвращать обратно пользователю:

```
def words_filter_handler(update,context):
    update.message.reply_text(bad_words_filter(update.message.text))
```

- Теперь мы определим эту функцию как обработчик сообщения, который будет вызываться, если он начинается со слова “filter” (с пробелом):

```
dp.add_handler(MessageHandler(Filters.regex('^filter '),words_filter_handler))
```

- Запустите программу. В телеграме напишите вашему боту сообщение, которое начинается со слова «filter» и далее любой текст, содержащий разные слова, в том числе и фильтруемые
- Проанализируйте ответ вашего бота. Плохие слова должны быть заменены «***»

3. Бот пишет числа прописью

Если в сообщении переданному нашему боту содержится только число, то пусть он напишет это число буквами, например 1 – «one», 10- «ten», 21 - «twenty one»

- Определите функцию num_to_words(). В качестве параметра эта функция принимает целое число, принятое ботом в сообщении от пользователя и возвращает строку содержащую указанное число прописью
- Определите функцию num_to_words_handler() , которая будет вызывать функцию перевода числа в пропись:

```
def num_to_words_handler(update,context):
    update.message.reply_text(num_to_words(int(update.message.text)))
```

- Теперь добавим эту функцию как обработчик сообщения, принятого ботом от пользователя. Эта функция будет вызываться, если в сообщении будут содержаться только символы от 0 до 9 и ничего более:

```
dp.add_handler(MessageHandler(Filters.regex('^[0-9]*'),num_to_words_handler))
```

- Запустите программу. Отправляйте в телеграмме вашему боту числа 12,23,98 и т.д. Проанализируйте ответ бота

Теперь наш бот умеет переводить числа в пропись!

4. Бот вычисляет математические выражения

Научим бот решать математические задачи. На вход подается строка с арифметическим выражением. Бот должен выполнить простейшие операции: сложение, умножение и деление и вернуть результат. Допускаются разные выражения, например, получив строку, содержащую «23+2*2» функция должна вернуть целое число 27. Для упрощения задачи мы не используем в скобки и будем работать только с целыми числами.

- Определите функцию calc_expression(). Функция в параметре получает строку, содержащую арифметическое выражение, и возвращает целое число как результат вычисления, либо слово «error»
- Определите функцию calc_expression_handler(), которая будет передавать полученное арифметическое выражение от пользователя функции calc_expression для расчета:

```
def calc_expression_handler(update,context):
    update.message.reply_text(calc_expression(update.message.text))
```

- Теперь добавим эту функцию как обработчик сообщения, принятого ботом от пользователя. Эта функция будет вызываться, если в сообщении будут содержаться символы, соответствующие регулярному выражению, определяющим простейшее арифметическое выражение:

```
dp.add_handler(MessageHandler(Filters.regex(\
r'^(\d+[\+\-\*\/]\d+){1}([\+\-\*\/]+\d+)*$'),\
calc_expression_handler))
```

- Запустите программу. Отправляйте в телеграмме вашему боту арифметические выражения, содержащие только числа и знаки. Проанализируйте его ответы

5. Бот и теория графов

С помощью бота мы построим граф. Для этого мы будем создавать собственные типы данных. Нам необходим:

- Класс Vertex, представляющий вершину графа
- Класс Edge, представляющий направленное ребро, соединяющее две вершины графа. Таким образом, если мы хотим попасть из одной вершины в другую, а потом обратно, нам нужно использовать два ребра
- Класс Graph, который хранит список всех вершин графа, а также текущую вершину графа

5.1. Подготовительные шаги

- Определите метод work_with_graph(), который в качестве параметра получает строку сообщения, а возвращает ответ, который надо передать пользователю. Сейчас верните в этом методе любую текстовую строку
- Определите метод work_with_graph_handler(), который вызывается, когда бот получит сообщения от пользователя, содержащую одну из следующих строк (звездочкой показано любые символы, любого количества, кроме пробелов):
    - add vertex * - добавляет вершину с указанным именем в граф
    - add edge * * - соединяет две вершины направленным ребром
    - go * - перейти к указанной текущей вершине, при условии, что она присоединена к текущей одним из ребер
- Добавьте вызов обработчика диспетчеру телеграм-бота:

```
dp.add_handler(MessageHandler(Filters.regex('^add|go'),work_with_graph_handler))
```

- Запустите программу. Проверьте, что на вышеописанные команды возвращается результат функции work_with_graph()

5.2. Определяем классы графа

- Создайте файл graph.py
- Определите в этом файле класс Vertex, реализующий вершину графа, со следующими методами (пока можно без операторов, указав ключевое слово pass, в следующей строке, после объявления метода):
    - __init__() – конструктор, который принимает в качестве параметра имя вершины. Имя вершины следует запомнить внутри объекта
    - get_name() – метод возвращает имя вершины
    - get_edge_for_connected_vertex() – метод, который возвращает ребро, соединяющее вершину, для которой этот метод был вызван с вершиной, чье имя было передано в качестве параметра. Если такое ребро отсутствует, должно возвращаться None
- Определите в файле graph.py класс Edge, реализующий направленное ребро графа, со следующими методами (пока можно без операторов, указав ключевое слово pass, в следующей строке, после объявления метода):
    - __init__() – конструктор, который принимает в качестве параметра вершину, к которой присоединяется ребро (вершина, на которую ребро направленно)
    - get_connected_vertex() – метод, возвращающий присоединенную вершину к ребру
- Определите в файле graph.py класс Graph, реализующий граф, со следующими методами (пока можно без операторов, указав ключевое слово pass, в следующей строке, после объявления метода):
    - __init__() – конструктор, который инициализирует пустым список вершин графа, а также определяет поле текущий вершины, присваивая ему значение None
    - add_vertex() – метод добавления вершины в список вершин графа. В параметре метод принимает добавляемую вершину
    - find_vertex() – метод ищет вершину по имени, которое передается в параметре метода, в списке вершин графа и возвращает ссылку на нее, либо None, если вершина не найдена
    - add_edge() – метод добавления ребра графа. В параметрах метода принимается имя исходной вершины и имя присоединенной вершины. Метод должен сначала найти исходную вершину в графе, затем вершину, к которой ребро будет направлено, а в конце создать ребро и связать эти две вершины

- set_current_vertex() – метод, который осуществляет переход от текущей вершины к вершине по имени, переданным в качестве параметра метода, через ребро графа, соединяющего текущую вершину с этой вершиной. Если переход возможен, метод устанавливает новую текущую вершину и возвращает на нее ссылку. Если же переход не возможен (ребро не соединяет текущую вершину с требуемой) текущая вершина не меняется и метод возвращает None

5.3. Реализация графа

- Определите все методы классов, согласно их спецификации в пункте 5.2
- В начало файла с вашим ботом добавьте импорт модуля с нашим графом:

```
import graph
```

- Перед методом work_with_graph определите глобальную переменную, в которой сохраните созданный объект графа (обратите внимание, что обращение к классу граф выполняется с указанием названия модуля, который вы ранее импортировали командой «import»):

```
bot_graph = graph.Graph()
```

- В методе work_with_graph() определите какую команду передал нашему боту пользователь и вызовите соответствующий метод. Совет: вы можете использовать метод split() для строки сообщения, чтобы получить список слов из которых состоит команда, после чего используя оператор if elif else организовать необходимую логику метода. Список команд приведен в пункте 5.1
- Запустите программу. Отправляйте из телеграмма вашему боту команду и проанализируйте ответа.

Вы написали свой первый бот на Python!