**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Кафедра “фундаментальная информатика и информационные технологии”**

**отчет**

**по домашнему заданию “  Функциональный Чат-Бот ”**

**по дисциплине «Функциональное программирование»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 22Б16 |  | Олизько С.С. |
| Преподаватель |  | Киямов Ж.У. |

**Санкт-Петербург**

**2023 г.**

**Оглавление**

1. Цель работы
2. Задача
3. Теоритическая часть
4. Алгоритм метода
5. Описание программы
6. Рекомендации по использованию
7. Вывод

**Цель:**

Создать функционального чат-бота, способного обрабатывать запросы пользователя, принимать решения на основе функциональных правил и предоставлять соответствующие ответы.

**Задача:**

Разработать механизм обработки текстовых запросов от пользователя. Учесть разнообразные типы запросов, такие как вопросы, команды и запросы на получение информации. Бот должен поддерживать состояния. Добавить возможность логирования важных событий или запросов для обеспечения отслеживаемости работы бота.

**Теоретическая часть:**

Библиотека telebot — это мощный инструмент для создания чат-ботов для мессенджера Telegram, написанный на языке программирования Python. Она предоставляет удобный и интуитивно понятный интерфейс для взаимодействия с Telegram Bot API, что делает процесс разработки чат-ботов быстрым и эффективным. Telebot обеспечивает простое подключение к API Telegram, а также предоставляет различные возможности для обработки входящих сообщений, команд, и событий.

Одной из ключевых особенностей telebot является поддержка обработки разнообразных типов сообщений, включая текстовые сообщения, изображения, аудио и документы. Библиотека также обладает механизмами для создания интерактивных клавиатур, обработки команд и поддержки состояний беседы, что позволяет разработчикам создавать более сложные и интересные взаимодействия с пользователями.

Telebot активно поддерживается и обновляется сообществом разработчиков, что обеспечивает его совместимость с последними изменениями в API Telegram и добавление новых функциональных возможностей. Благодаря своей простоте в использовании и богатому функционалу, библиотека telebot стала популярным выбором для создания чат-ботов в экосистеме Telegram.

При обсуждении библиотек для работы с API в контексте программирования, важно отметить, что существует множество библиотек, предназначенных для упрощения взаимодействия с внешними интерфейсами приложений (API). Одной из широко используемых библиотек в этой области является библиотека requests для языка программирования Python.

Библиотека requests обеспечивает простой и элегантный способ отправки HTTP-запросов к веб-ресурсам и обработки полученных ответов. Её простой синтаксис делает возможным легкое взаимодействие с различными API, позволяя программистам получать данные, отправлять данные и выполнять другие HTTP-операции с минимальными усилиями. Это делает requests популярным выбором для создания клиентов API в различных проектах, включая чат-боты, веб-приложения и многое другое.

**Алгоритм метода:**

1. **Инициализация библиотек и токена:**
   * Импорт необходимых библиотек, таких как telebot и requests.
   * Установка токена Telegram Bot API.
2. **Определение глобальных переменных и словарей:**
   * Создание словаря user\_names для хранения имен пользователей.
   * Установка флага flg\_weath в значение False для управления процессом запроса о погоде.
3. **Реализация функции логирования:**
   * Определение функции log\_event(event), которая записывает событие в файл "log.txt".
4. **Реализация функции получения погоды:**
   * Определение функции get\_weather(city), которая отправляет запрос к API OpenWeatherMap для получения информации о погоде в указанном городе.
5. **Реализация функции получения курсов валют:**
   * Определение функции get\_currency\_exchange\_rates(), которая отправляет запрос к API Open Exchange Rates для получения курсов основных валют к рублю.
6. **Реализация функции получения шутки Chuck Norris:**
   * Определение функции get\_chuck\_norris\_joke(), которая отправляет запрос к API Chuck Norris Jokes для получения случайной шутки.
7. **Реализация функции получения имени пользователя:**
   * Определение функции get\_user\_name(user\_id), которая возвращает имя пользователя из словаря user\_names.
8. **Реализация функции сохранения имени пользователя:**
   * Определение функции save\_user\_name(user\_id, name), которая сохраняет имя пользователя в словаре user\_names.
9. **Обработка команды /start:**
   * Определение хэндлера команды /start, который отправляет приветственное сообщение и запрашивает имя пользователя, если оно еще не сохранено.
10. **Обработка введенного имени пользователя:**

* Определение хэндлера для обработки введенного имени пользователя после команды /start.

1. **Обработка команды /help:**

* Определение хэндлера команды /help, который отправляет справочное сообщение о функционале бота.

1. **Обработка текстовых сообщений от пользователя:**

* Определение хэндлера для обработки текстовых сообщений.
* Проверка содержания ключевых слов в тексте для определения необходимого действия (получение погоды, курсов валют или шутки).

1. **Запуск бота:**

* Запуск бота в режиме ожидания новых сообщений.

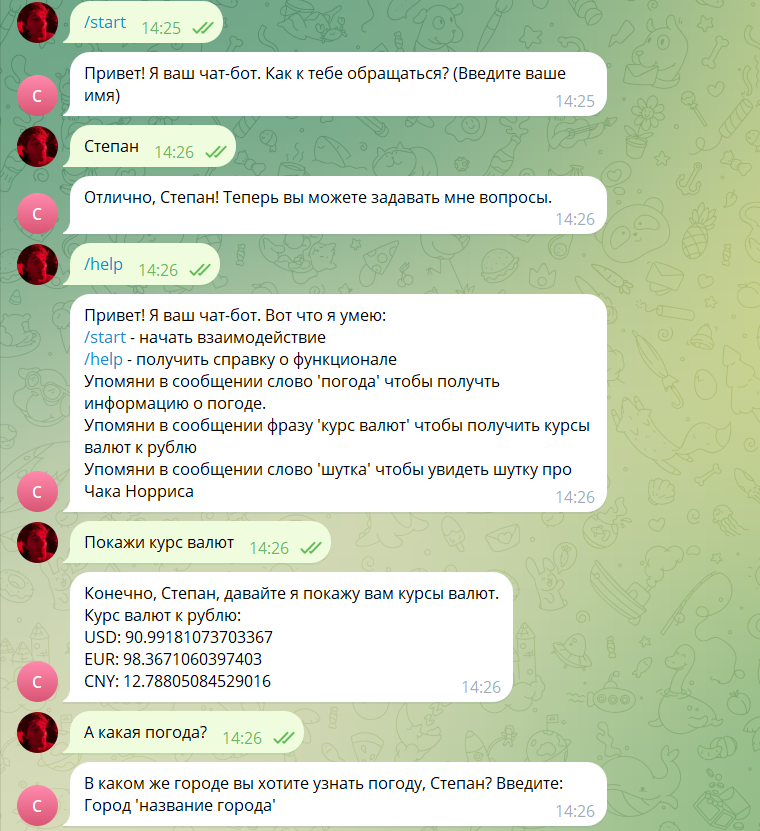
**Описание программы:**

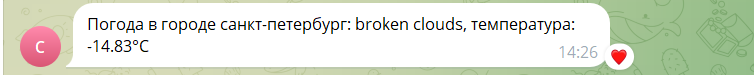
|  |  |  |
| --- | --- | --- |
| **Имя функции** | **Тип возвращаемого значения** | **Описание функции** |
| log\_event | None | Записывает событие в файл "log.txt". |
| get\_weather(city) | String | Получает информацию о погоде в указанном городе с использованием API OpenWeatherMap. Возвращает строку с данными о погоде. |
| get\_currency\_exchange\_rates | String | Получает курсы валют к рублю с использованием API Open Exchange Rates. Возвращает строку с данными о курсах валют. |
| get\_chuck\_norris\_joke | String | Получает случайную шутку про Чака Норриса с использованием API Chuck Norris Jokes. Возвращает строку с шуткой. |
| get\_user\_name(user\_id) | String | Возвращает имя пользователя по заданному идентификатору пользователя. |
| save\_user\_name(user\_id, name) | None | Сохраняет имя пользователя в словаре user\_names по заданному идентификатору пользователя. |
| handle\_start(message) | None | Обрабатывает команду /start. Если имя пользователя не сохранено, предлагает ввести его. |
| handle\_user\_name(message) | None | Обрабатывает введенное имя пользователя после команды /start. Сохраняет имя пользователя. |
| handle\_help(message) | None | Обрабатывает команду /help. Отправляет справочное сообщение о функционале бота. |
| handle\_text(message) | None | Обрабатывает текстовые сообщения пользователя. Проверяет ключевые слова и выполняет соответствующие действия. |

**Рекомендации по использованию:**

* **Начните с команды /start:** Для начала взаимодействия с чат-ботом, отправьте команду /start. Это позволит вам получить приветственное сообщение и, при необходимости, ввести ваше имя.
* **Укажите свое имя:** Если при запуске бота вас попросят ввести имя, не стесняйтесь указать его. Это поможет боту обращаться к вам более персонально.
* **Используйте команду /help:** Если вам нужна информация о функционале бота, отправьте команду /help. В ответ вы получите справочное сообщение с описанием доступных команд и возможностей.
* **Задавайте вопросы о погоде:** Если вас интересует погода в каком-то конкретном городе, упомяните слово "погода" в вашем сообщении, а затем укажите название города. Бот предоставит вам актуальную информацию о погоде.
* **Узнавайте курсы валют:** Если вам интересны текущие курсы валют к рублю, упомяните фразу "курс валют" в вашем сообщении. Бот предоставит вам информацию о курсах основных валют.
* **Хотите шутку? Упомяните Чака Норриса:** Если вы настроены на шутки, упомяните слово "шутка"

**Контрольный пример:**



****

**Вывод:**

Предоставленный код реализует чат бота чат-бота, способного обрабатывать запросы пользователя, принимать решения на основе функциональных правил и предоставлять соответствующие ответы.

**Список литературы:**

* <https://pytba.readthedocs.io/ru/latest/index.html>
* <https://docs.python.org/3/requests/os.html>

**GitHub:**

https://github.com/StephanOlizko/fp\_homework