**YourTelebot. Документация**

Оглавление

Введение 2

Изменение темы 2

Виджеты 3

Переменные 4

Функции 6

Теги 7

Структура проекта 9

Атрибуция, лицензия 10

Добавление своего языка 11

Добавление своей команды 12

Итог 17

**Введение**

Приветствуем в документации YourTelebot. Наше приложение написано на языке программирования python, использует множество библиотек и переменных. Данный мануал создан для того, чтобы помочь вам разобраться в коде и кастомизировать что-то.

Для начала, рекомендуем установить среду разработки, если она не установлена на вашем компьютере. Скачайте Visual Studio Code, Sublime Text, PyCharm или любой другой удобный для вас редактор кода. Не используйте стандартный блокнот для работы с python, это не рационально.

Данная документация не предусматривает подробнейшего объяснения всего кода. Тут нет объяснения, зачем нужна та или иная строка или библиотека. Большинство, чего нет в данной документации, можно свободно найти в интернете, является интуитивно понятным или объяснено прямо в коде с помощью комментариев.

Также данная документация написана с расчётом, что вы знаете основы python. Понимаете, что такое переменные, функции, словари, списки, зачем нужны некоторые библиотеки и прочие моменты.

Инструкция может быть актуальна только для [официальной версии приложения](https://github.com/qpikzz/YourTelebot).

Авторы модификаций имеют право менять внутренности программы, для удобства пользователей советуем указывать изменения в документации.

Изменение темы

В данном разделе мы покажем, как изменить цвет десктопного окна.

1. Откройте YourTelebot.pyw с помощью установленного ранее редактора кода.
2. Нажмите CTRL+F (найти) и впишите “Colors”. Таким образом, вы сможете найти 6 переменных (black, darkBlue, darkGreen… white). Внутри кавычек, вы можете вставить любой цвет в формате HEX кода (пример: #FFFFFF, #000013)

Смена шрифта является не желательной. Размер некоторых виджетов в tkinter привязан к используемому шрифту, поэтому приложение может “поплыть”.

Виджеты

В данном разделе мы объясним, за что отвечает тот или иной элемент окна десктопной версии.

**Токен** - сюда вы должны вставить API token телеграм бота, который вы создали в BotFather (подробнее в разделе «установка» на сайте с помощью).

**Пользователи** - в данный объект вы должны вставить id аккаунтов, которые получат доступ к вашему пк. Подробнее об этом также написано в разделе «установка» сайта с помощью.

**Скрывать данные под спойлером** - в случае, если данная опция включена, бот будет автоматически помещать под спойлер важные данные. Чтобы увидеть их нужно будет нажать на скрытую строку.

**Режим «2 сообщения»** - режим работы, в котором бот удаляет старые сообщения, оставляя в чате 2-3 сообщения. Это может быть удобно для того, чтобы удалять сразу все полученные данные и не засорять чат.

Из-за оптимизации может работать не идеально, поэтому в случае возникновения неполадок, вы можете всё поправить с помощью команды clearchat.

**Сохранять логи** - программа создаст файл log.txt во временной папке (по умолчанию temp), в который будут вписываться все действия пользователей.

**Язык** - настройка, позволяющая выбрать предпочтительный вариант для приложения. Все доступные переводы находятся в папке tl. Далее будет приведена инструкция по созданию собственного языка.

**Автосохранение файлов** - все фото, видео, кружочки, голосовые и документы, отправленные боту, будут сохранятся в папку temp.

**Консоль** - самый большой виджет в правой части программы. Как указано в приветственном сообщении, он предназначен для вывода различных сообщений, таких как: включение/выключение бота, использование команд. Вы можете очистить её с помощью команды “clear”.

**Включить/Выключить** - кнопка, весь функционал которой заключается во включении и выключении бота.

**Помощь** - открыть сайт с помощью. В данном месте вы сможете найти инструкцию по установке приложения, контакты поддержки, ответы на самые популярные вопросы и многое другое.

Переменные

Ниже мы расскажем о основных переменных.

**data** - основная переменная, представляющая из себя словарь. В ней содержатся настройки и все языки.

Структура:

|  |  |
| --- | --- |
| **Название** | **Тип** |
| settings | Dict (Словарь) |
| lang1 | Dict (Словарь) |
| lang2 | Dict (Словарь) |
| langN | Dict(Словарь) |

Вместо lang1, lang2, …, langN будут названия папок с переводами. В нашем случае, эти слова указаны лишь для примера.

Вы можете обратиться ко всем настройкам при помощи следующей конструкции:

|  |  |
| --- | --- |
| 1 | data["settings"] |

К примеру, выведем все настройки и их значение:

|  |  |
| --- | --- |
| 1 | print(data["settings"]) |

Это выдаст большую строку, в которой может быть сложно разобраться. Посмотрим на неё в более простом виде:

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **По умолчанию** |
| language | str | “en” |
| token | str | “” |
| users | list (str) | [] |
| botWorked | bool | false |
| groups | bool | false |
| answerWithPic | bool | true |
| hide | bool | false |
| log | bool | false |
| 2mesMode | bool | false |
| autoSave | bool | false |
| tempFolder | str | “temp” |

Давайте поясним некоторые пункты, содержимое которых может быть не очевидно:

1. botWorked - вспомогательная переменная, определяющая, работает бот или нет.
2. groups - скрытая настройка, позволяющая включить бота в группах. Не рекомендуем этого делать, ведь в вашу беседу может проникнуть другой человек и узнать данные, которые не должен был знать.
3. answerWithPic - ответ с картинкой. Это означает, что бот будет прикреплять скриншот туда, где это хоть каплю уместно. К примеру: в командах write, click и т.п.
4. hide - скрыть под спойлером.
5. autoSave - автоматическое сохранение отправленных боту картинок, файлов, голосовых и кружочков.
6. tempFolder - временная папка по умолчанию.

Приведём пример просмотра и изменения настройки hide.

|  |  |
| --- | --- |
| 1 | print(data["settings"]["hide"]) *# Вывод: False* |
| 2 | data["settings"]["hide"] = not data["settings"]["hide"] *# Меняем на противоположное значение* |
| 3 | print(data["settings"]["hide"]) *# Вывод: True* |

Все «ключи» в data можно вывести, с помощью приведённого ниже кода. Это поможет узнать все доступные языки.

|  |  |
| --- | --- |
| 1 | print(data.keys()) |
| 2 | *# Вывод: dict\_keys(['settings', 'en', 'ru'])* |

Разберём состав языков:

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Краткое описание** |
| error | dict | Сообщения об ошибках |
| log | dict | Сообщения, вписываемые в log.txt |
| messages | dict | Отправляемые ботом ответы |
| system | dict | Текст виджетов, десктопного окна и т.п. |

Если вам нужно узнать какое-либо сообщение или заменить его значение - откройте его файл-категорию в папке tl/(язык)/(пункт).txt. Для того, чтобы редактор кода красиво подсвечивал синтаксис включите язык «JSON».

**tk** - переменная, в которой хранится окно программы. К ней применимы все функции библиотеки tkinter, пример есть ниже.

|  |  |
| --- | --- |
| 1 | tk["bg"] = black *# Установить задний фон (black = #28262A)* |
| 2 | tk.title("YourTelebot") *# Установить заголовок приложения* |

**users, token, ..., twoMesMode** - виджеты, в которых хранится актуальная информация о настройках приложения. Конечно, вы можете получить эти данные из data[“settings”], но не будет лишним знать о том, как получить их напрямую.

|  |  |
| --- | --- |
| 1 | token.get() *# Получить токен бота, в  виде строки.* |
| 2 | users.get("1.0","end-1c").split("\n") *# Получить список пользователей.* |
| 3 | hideSpoiler.get() *# Получить булевое значение настройки «скрыть под спойлером»* |
| 4 | *# Все переменные подобного рода: token, users, hideSpoiler, twoMesMode, logs, autoSave, language* |

**consoleMessage** - переменная, хранящая текст, который необходимо вывести в консоль. Чтобы учесть всё (переносы строки, вставку тегов и другое), рациональнее использовать функцию “writeInConsole(text)”, но об этом мы поговорим позже.

**user, text, chat** - вспомогательные переменные в функциях start, text и media. Используются для сокращения кода. К примеру, вместо указания message.from\_user.id (вывод id пользователя), можно ввести только user.id.

Значение остальных переменных на наш взгляд является интуитивно понятным.

Функции

В данном разделе будет рассказано о назначении некоторых функций. В скобках указаны аргументы, которые она принимает.

**writeInConsole(***text***)** - как было указано ранее, данная функция нужна для вставки текста в консоль. Является отдельным потоком (работает без зависимости от основного окна). Это необходимо, для исправления некоторых недочётов библиотеки tkinter.

**openData()** - функция, для создания переменной data. Нужна для прочтения настроек и переводов.

**answerWithPic(***message, text***)** - ответ бота с скриншотом. Аргумент “message” - сообщение, получаемое ботом. Приведем пример:

|  |  |
| --- | --- |
| 1 | bot=telebot.TeleBot(data["settings"]["token"],parse\_mode="html") |
| 2 | @bot.message\_handler(commands=["start"]) |
| 3 | def start(message): |
| 4 | answerWithPic(message, "text") |

**mode2mes(***bot, message***)** - функция, обеспечивающая работу режима с двумя сообщениями. Для оптимизации, она работает так: перебирает все сообщения с последнего до первого и удаляет их. В случае, если 7 сообщений подряд были удалены до этого, она решает, что чат был очищен, а значит дальше перебирать нет смысла.

**botFunc()** - как и консоль, является отдельным потоком. Хранит в себе весь код telegram бота. Также, данная функция botFunc имеет 3 вложенные функции: **start(***message***)**, **text(***message***)** и **media(***message***)**. Если вы знакомы с модулем telebot, их назначение вам понятно, но мы всё равно кратко поясним. Функция “start” содержит в себе код, исполняемый тогда, когда пользователь отправил команду “/start”. “text” же в содержит код, исполняемый при отправке обычных текстовых сообщений. В match (аналог свичкейса из C-подобных языков) можно найти почти все команды. Чуть позже мы рассмотрим, как создавать свои. Функция “media”, как понятно из названия, работает с файлами, голосовыми, фото, видео и так далее.

**botStartStop()** - функция, из названия которой понятно, что она нужна для переключения состояния бота. Помимо изменения значения botWorking настроек, меняет вид кнопки Включить/Выключить, а также выводит сообщение в консоль.

**log(***text, message***)** - функция для сохранения логов. Передача “message” нужна для тегов (вставка имени и id пользователя, текста и прочее).

**help()** - вызов сайта с помощью. Нужна для определения темы - если на данный момент меньше 6 или больше 18 часов, то вызовется тёмная, иначе светлая.

**saveSettings()** - автоматическое сохранение настроек. Было решено отказаться от стандартной кнопки «Сохранить» для увеличения свободного места в окне и большего удобства пользователей. Является отдельным потоком.

**setTag(***text, message***)** - особо внимательные пользователи могли заметить, что несмотря на то, что данная функция стоит в начале кода, мы объясняем её в конце раздела. Это связано с тем, что она напрямую связана со следующим разделом. Данная функция меняет теги приложения (вписываются в квадратных скобках) на html теги. Это необходимо для форматирования сообщений (жирный шрифт, курсив и так далее) и вставки некоторых переменных в текст. Если используется вне дочерних функций botFunc, вместо message необходимо указывать “None”.

Остальные функции являются вспомогательными для виджетов, их изменение не желательно, поскольку может сломать некоторые элементы окна. Если вам нужен пример использования какой-либо функции, вы сможете найти его непосредственно в коде приложения.

Теги

Просматривая файлы с переводами, вы могли заметить странные конструкции, взятые в квадратные скобки. Это и есть теги. Они нужны для оптимизации разработки языка и вставки переменных. К примеру, вместо <blackquote> (цитата), можно использовать [bq].

Выделяется 3 типа тегов:

1. **Постоянный** - можно использовать в любом месте кода в любое время .
2. **Зависимый** - выполнение зависит от некоторой переменной. К примеру, чтобы использовать теги, зависящие от message, требуется указать данную переменную в setTag.
3. **Локальный** - используется только в определённом месте.

Некоторые теги имеют альтернативные версии. Другими словами, не имеет разницы, впишите вы [b] или [bold], в любом случае это заменится на <b>.

**Постоянные:**

|  |  |  |
| --- | --- | --- |
| **Тег** | **Альт\*** | **Краткое описание** |
| b | bold | Жирный текст. **Вот такой.** |
| blackquote | bq | Цитата. |
| code | cod | Переделать текст в код. Можно скопировать нажатием на текст. |
| date | - | Текущая дата в формате “День.Месяц.Год”. |
| h | hide | В случае, если у пользователя включена настройка «скрывать данные под спойлером» будет прятать указанное внутри, иначе равносильно [code] |
| i | italic | Курсив. *Вот такой.* |
| ip | - | Вставка ip адреса компьютера, на котором запущена программа. |
| n | - | Перенос строки. |
| now | - | Текущее время, с точностью до миллисекунды. |
| tempFolder | - | Актуальная временная директория. |
| time | - | Настоящее время в формате “Час:Мин:Сек”. |
| spoiler | sp | Скрыть текст под спойлером |
| u | underline | Подчёркивание. Вот такое. |
| x | - | Расположение курсора по оси X |
| y | - | Расположение курсора по оси Y |

\*Альт - сокр. Альтернативный

**Зависимые:**

|  |  |  |
| --- | --- | --- |
| **Тег** | **Перемен\*** | **Краткое описание** |
| user\_username | message | Никнейм пользователя. Пример: @durov |
| user\_id | message | ID пользователя. Пример: 123456789 |
| user\_first\_name | message | Имя пользователя, указанное в телеграм. |
| user\_last\_name | message | Фамилия пользователя. |
| bot\_username | bot | Никнейм бота, токен которого указан в приложении. |
| bot\_id | bot | ID вашего бота. |
| location | otherInfo | Локация, определяемая по ip. Может работать не стабильно. |
| org | otherInfo | Провайдер интернета, который используется на вашем компьютере. |
| timezone | otherInfo | Часовой пояс, определяемый по ip. |

\*Перемен - сокр. переменная, от наличия которой зависит выполнение тега.

Остальные теги, которые вы встретите являются локальными, и вероятнее всего работают только в данной строке.

**Добавление собственного тега** - при расширении возможностей бота у вас может появиться необходимость в добавлении своих тегов.

Если он является постоянным, нажмите ctrl+F и введите «Tags», затем в словаре replaces вставьте ваше изменение в формате json (“[тег]”:”<tag>”)

В случае, если обновление является зависимым, после объявления словаря с постоянными тегами вставьте данную конструкцию:

|  |  |
| --- | --- |
| 1 | if value: |
| 2 | replaces.update({ |
| 3 | "[Tag]":"<Tag>" |
| 4 | *# “[value]”:value* |
| 5 | }) |

Где value - переменная, наличие которой необходимо для выполнения, [Tag] - ваш тег, <Tag> - новый текст или переменная.

Если ваш тег будет встречаться только в 1 месте, к примеру в определённой команде, можно использовать данный метод:

|  |  |
| --- | --- |
| 1 | *# Вырвано из команды “create”:* |
| 2 | case "create": |
| 3 | with open(file,"w") as createFile: |
| 4 | bot.send\_message(chat.id,setTag(data[lang]["messages"]["create"],message).replace("[file]",file)) |
| 5 | *# [file] меняется на адрес файла (к примеру: temp/hello.txt).* |

Структура проекта

Сейчас мы подробнее расскажем о том, где и какие файлы находятся.

Начало пути - корневая папка проекта. Содержимое:

1. **yourTeleBot.pyw** - основной файл. В нём и хранятся кости приложения.
2. **data** - данные, необходимые для работы программы.
3. **temp** - временная директория. Может иметь другое название или не существовать вовсе.
4. **tl** - английское сокращение от «translates», что означает «переводы». Хранит в себе все языки.

Спускаемся сверху вниз, проходясь по каждой папке. Перемещаемся в **data**.

1. **.ttf файлы** - шрифты. Вы должны были скачать их на стадии установки.
2. **settings** - хранит значение всех настроек, изложенных в разделе «Переменные».
3. **splashes.txt** - файл, содержащий все заголовки десктопной версии. Находятся в левом верхнем углу при запуске приложения.
4. **images** - изображения. В основном иконки программы, а также пара изображений, используемых внутри десктопной версии. Разрешение глаз: 20х13 пикселей.

Содержимое **temp** крайне непостоянно, но вот, что там можно встретить:

1. **log.txt** - с точностью до секунды фиксирует кто и что использовал.
2. **voice.mp3** - речь, синтезированная при помощи команды “say”
3. **photo.jpg** - фотография, полученная с вашей вебкамеры с помощью команд “photo” и “colorphoto”.
4. **Прочее** - сохранённые медиа, либо созданные при помощи “create” файлы, у которых не указали директорию.

Внутренности **tl** представляют из себя ещё несколько папок. По умолчанию это: en, none и ru. Как можно догадаться, каждая из них хранит какой-либо перевод. Потроха у них всех одинаковые:

1. **dark.html** и **light.html** - веб страница с помощью. Открывается локально в вашем браузере.
2. **error.txt** - файл, хранящий реакции на различные ошибки.
3. **log.txt** - сообщения, вписываемые в логах.
4. **manual.docx** - то, что вы читаете прямо сейчас.
5. **messages.txt** - отправляемые ботом реплики, в зависимости от команды.
6. **system.txt** - фразы, используемые в интерфейсе десктопной версии.

Не сложно заметить, что все файлы .txt на самом деле имеют json формат.

Атрибуция, лицензия.

Вероятнее всего, вы уже поняли, что YourTelebot распространяется под лицензией **CC BY 4.0**. Давайте разберём, что это значит.

Если кратко, вы можете:

1. Делиться (обмениваться) - копировать и распространять материал как угодно и в любом формате (включая персональное и коммерческое использование)
2. Адаптировать (создавать производные материалы) - видоизменять и создавать новое, опираясь на этот материал. В том числе в коммерческих целях.

ПРИ ОБЯЗАТЕЛЬНОМ СОБЛЮДЕНИИ СЛЕДУЮЩИХ УСЛОВИЙ:

1. Атрибуция – вы должны обеспечить соответствующее указание авторства. Вы можете это делать любым разумным способом, но не таким, который подразумевал бы, что создатель приложения одобряет вас или ваш способ использования произведения. Рекомендуем вставить атрибуцию в приветствующем тексте консоли или на веб странице с инструкциями. Шаблон для html (в веб страницу) и обычный текст будут предоставлены позже.
2. Без дополнительных ограничений – вы не в праве применять юридические ограничения или технологические меры, создающие другим юридические препятствия в выполнении чего-либо из того, что разрешено лицензией.

Подробнее здесь: <https://creativecommons.org/licenses/by/4.0/>

**Атрибуция**

HTML – список изменений указывать в 2 строке (которая пустая), другими словами в теге <p> с id = “changes”.

|  |  |
| --- | --- |
| 1 | <div style="border: 1px solid #DADBDF; padding: 3rem 0; text-align: center; font-size: 1.5rem; color: #02000A; background-color: #E8E6EA"><p><a style="text-decoration: none; color: #2E5D42" href="https://github.com/YourTelebot">YourTelebot</a> © 2024 by <a style="text-decoration: none; color: #2E5D42" href="https://t.me/qpikzzbot">qpikzz</a> is licensed under <a style="text-decoration: none; color: #2E5D42" href="https://creativecommons.org/licenses/by/4.0/">CC BY 4.0</a></p><p style="font-size: 1rem;">differences from the original version:</p><p style="font-size: 1rem; color: #2E5D42;" id="changes"> |
| 2 |  |
| 3 | </p></div> |

Текст - можно вставить куда угодно. Мы рекомендуем в system > consoleTitle. Вместо [changes] указать внесённые вами изменения. Также советуем перевести эту надпись для всех языков.

|  |  |
| --- | --- |
| 1 | [YourTelebot](https://github.com/YourTelebot) © 2024 by [qpikzz](https://t.me/qpikzzbot) is licensed under [CC BY 4.0](https://creativecommons.org/licenses/by/4.0/). Differences from the original version: [changes] |

Добавление своего языка

С теорией покончено. Можем переходить к реальным действиям. Для начала создадим новый язык.

1. Перейдите в директорию tl и дублируйте папку none. Это стандартный шаблон, в нём все поля пустые для удобства заполнения. Переименуйте полученную копию так, как хотите назвать ваш язык. Рекомендуется использовать не более 5-6 букв.
2. Откройте созданную папку. Для того, чтобы точно ничего не забыть и не перепутать, советуем параллельно открыть какой-либо существующий перевод.
3. Заполните .txt файлы, ни одна из строк не должна быть пуста, иначе это может сломать приложение.
4. После того, как вы заполните все пропуски, уже можно перезапустить приложение и выбрать ваш язык. Если вы не хотите распространять созданный вами перевод и ещё больше заморачиваться, этого может быть достаточно, в ином случае приступаем к редактированию .html файлов, документации и пользовательского соглашения.
5. Для редактирования веб страницы с помощью будет неплохо, если вы знаете html и css. Внутри тега <body>, в дочерние <a>, <p> и прочие вставьте ваши переводы. Крайне рекомендуем операться на уже существующие версии.
6. В документации оставлены лишь части, актуальные для любого языка, а именно: таблицы, вставки кода (без комментариев), нумерованные списки. Ориентируясь на них вписывайте новые части.
7. Пользовательское соглашение не обладает особыми чертами, просто перепишите на вашем языке. Важно - вы не должны менять суть. Смысл изложенного должен быть сохранён.

Добавление своей команды

Приведём 2 примера: добавление «эхо» в нашего бота, а также создание команды «random», которая по итогу попадёт в финальную версию программы.

**Эхо**

Как вы могли заметить, в оригинальной версии этого нет. Это связано с тем, что не получив дальнейшего ответа пользователь, который случайно нашёл нашего бота с большей вероятностью просто уйдёт, но для примера разберём добавление этого.

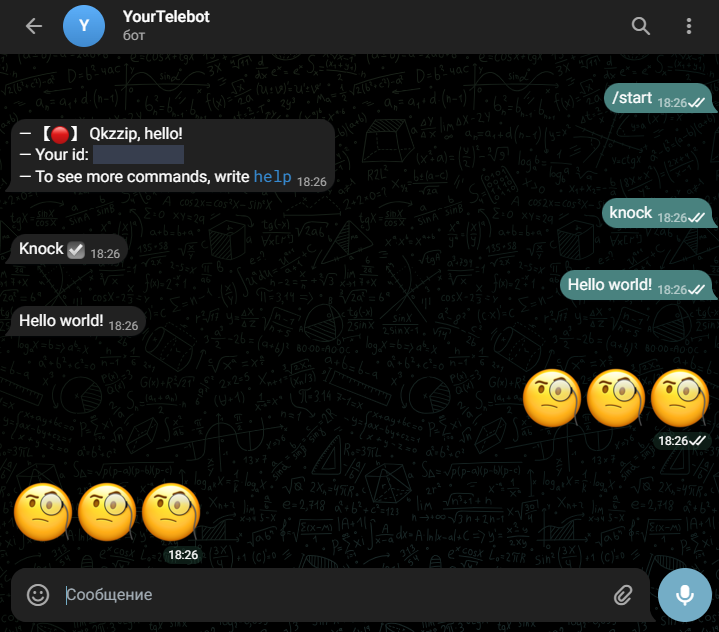
Более понятным языком - если бот не нашёл, что ответить на нашу команду, он просто отправит её нам заново.

Если бы бот вычислял, какую команду требуется исполнить при помощи if/elif, то мы бы просто использовали else, но в YourTelebot используется match. Альтернативой является “case \_:”, пропишем это после всех существующих команд.

Внутри «случая», пропишем отправку пользователю его же сообщения. Это можно реализовать с помощью TeleBot, функции bot.send\_message. Весь код занимает 2 строки и выглядит вот так:

|  |  |
| --- | --- |
| 1 | case \_: |
| 2 | bot.send\_message(chat.id, text) |

Проверим написанный код.



Отлично! Всё работает!

Переделаем это так, чтобы бот отправлял не сообщение пользователя, а уведомление о том, что он не понимает нас и не знает такой команды.

В первую очередь добавим саму фразу, для этого впишите её во все имеющиеся у вас переводы. Поскольку это сообщение, отправляемое ботом, рекомендуется использовать messages.txt:

|  |  |
| --- | --- |
| 1 | "whachaMean":"Что это значит??? Я не знаю такой команды :(" |

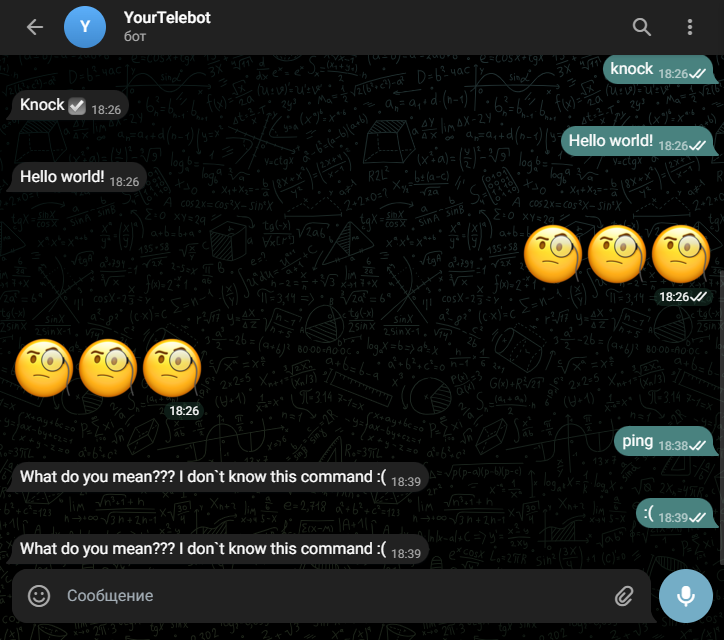
Далее внутри «bot.send\_message» заменим text на функцию setTag, в которую и передадим нашу фразу.

Конечно, сейчас мы не используем теги и может появиться вопрос, зачем использовать установку тегов? Это сделано на случай, если в будущем вы внезапно решите добавить больше форматирования или поменять текст так, что в него потребуется вставить какую-либо переменную.

Получившийся новый код:

|  |  |
| --- | --- |
| 1 | case \_: |
| 2 | bot.send\_message(chat.id,setTag(data[lang]["messages"]["whachaMean"], None)) |

Перезапустим бота и проверим изменения в этом же чате:



**Random**

Сделаем команду, которая будет возвращать случайное число. По умолчанию, если пользователь не вводил дополнительных чисел, она выдаст число в промежутке от 0 до 100. В случае, если указано 1 число (к примеру: random 10), то бот выдаст случайное число от 0 до 10. Когда указано 2 числа, возвращается случайное в указанном горизонте. Пример: если указать 1000 и 200, выберется число в диапазоне от 200, до 1000.

Для начала впишем необходимые нам фразы в переводы. Сделаем это в messages.txt.

|  |  |
| --- | --- |
| 1 | "randomDefault":"Случайное число от 0 до 100: [cod][rand][/cod]", |
| 2 | "randomWithA":"Случайное число от 0 до [A]: [cod][rand][/cod]", |
| 3 | "randomWithAB":"Случайное число от [A] до [B]: [cod][rand][/cod]" |

Первое сообщение будет использоваться тогда, когда пользователь не указал ни одного числа.

Второе будет отправляться в ответ на сообщение с 1 параметром.

Третье, очевидно, для случая, когда указаны оба конца промежутка.

Важный момент, тег [B] обязательно должен быть написан с большой буквы. Если написать с маленькой, будет ошибка, ведь он замениться на <b> (тег открытия для жирного текста).

Добавим «случай» в match, внутри создадим список с числами, которые указал пользователь:

|  |  |
| --- | --- |
| 1 | case "random": |
| 2 | rand = text.split() *# Создание списка со всеми словами в сообщении* |
| 3 | rand.pop(0) *# Удаление из него слова «random»* |
| 4 | rand = list(map(int, rand)) *# Переработка всех эллементов из str в int* |

Далее нам необходимо проверить, сколько элементов имеется в списке rand, от этого и будет зависеть ответ. Сначала будет логичным сделать случай, когда нет ни одного числа.

|  |  |
| --- | --- |
| 1 | if not rand: |
| 2 | bot.send\_message(chat.id, setTag(data[lang]["messages"]["randomDefault"], None).replace("[rand]",str(r.randint(0,100)))) |

Во 2 строке может быть не просто разобраться. Поясняем, сначала в наше сообщение randomDefault из messages.txt вставляются постоянные теги, а затем локальный тег [rand] меняется на случайное число в диапазоне от 0 до 100. После данных махинаций полученное сообщение отправляется пользователю.

Аналогично пропишем для случая, когда человек указывает только 1 число. Логика здесь проста – у вас есть коробка с 1 мячом, вас просят достать из коробки случайный мяч. Естественно это будет этот самый единственный мяч.

|  |  |
| --- | --- |
| 1 | elif len(rand) == 1: |
| 2 | bot.send\_message(chat.id, setTag(data[lang]["messages"]["randomWithA"], None).replace("[rand]",str(r.randint(min(0,rand[0]),max(0,rand[0])))).replace("[A]",str(rand[0]))) |

Ещё одно отличие от «стандартного» случая - локальный тег [A], заменяемый на введённое пользователем число. Далее появится такой же тег [B].

Добавим условие, когда пользователь ввёл другое количество цифр. Конечно, мы будем использовать только 2, но и ограничивать длину сообщения нам незачем. Остальные будут просто проигнорированы.

|  |  |
| --- | --- |
| 1 | else: |
| 2 | if rand[0] == rand[1]: |
| 3 | bot.send\_message(chat.id, setTag(data[lang]["messages"]["randomWithAB"], None).replace("[rand]",f"{rand[0]} ¯\\\_(ツ)\_/¯").replace("[A]",str(rand[0])).replace("[B]",str(rand[1]))) |
| 4 | else: |
| 5 | bot.send\_message(chat.id, setTag(data[lang]["messages"]["randomWithAB"], None).replace("[rand]",str(r.randint(min(rand[0],rand[1]),max(rand[0],rand[1])))).replace("[A]",str(min(rand[0],rand[1]))).replace("[B]",str(max(rand[0],rand[1])))) |

Обязательно не забываем про min() и max(). Если в тегах A и B это имеет чисто эстетический характер, то в randint может вызвать ошибку, поскольку в нём первым параметром обязательно должно идти меньшее значение, а далее большее.

Не забудем и про людей, которые на вопрос «сколько?» отвечают «много.» Если ввести вместо числа буквы, наша программа сломается. Чтобы этого не допустить покроем код, отслеживая ошибку ValueError.

Добавим дополнительную фразу для этого в error.txt

|  |  |
| --- | --- |
| 1 | "numberRandomError":"Вы указали не число![n]Пример: [cod]random 1 10[/cod]" |

Весь код команды выглядит так:

|  |  |
| --- | --- |
| 1 | case "random": |
| 2 | try: *# Отслеживаем ошибку* |
| 3 | rand = text.split() *# Создание списка со всеми словами в сообщении* |
| 4 | rand.pop(0) *# Удаление из него слова «random»* |
| 5 | rand = list(map(int, rand)) *# Переработка всех эллементов из str в int* |
| 6 |  |
| 7 | if not rand: |
| 8 | bot.send\_message(chat.id, setTag(data[lang]["messages"]["randomDefault"], None).replace("[rand]",str(r.randint(0,100)))) |
| 9 |  |
| 10 | elif len(rand) == 1: |
| 11 | if rand[0] == 0: |
| 12 | bot.send\_message(chat.id, setTag(data[lang]["messages"]["randomWithA"], None).replace("[rand]","0 ¯\\\_(ツ)\_/¯").replace("[A]","0")) |
| 13 | else: |
| 14 | bot.send\_message(chat.id, setTag(data[lang]["messages"]["randomWithA"], None).replace("[rand]",str(r.randint(min(0,rand[0]),max(0,rand[0])))).replace("[A]",str(rand[0]))) |
| 15 |  |
| 16 | else: |
| 17 | if rand[0] == rand[1]: |
| 18 | bot.send\_message(chat.id, setTag(data[lang]["messages"]["randomWithAB"], None).replace("[rand]",f"{rand[0]} ¯\\\_(ツ)\_/¯").replace("[A]",str(rand[0])).replace("[B]",str(rand[1]))) |
| 19 | else: |
| 20 | bot.send\_message(chat.id, setTag(data[lang]["messages"]["randomWithAB"], None).replace("[rand]",str(r.randint(min(rand[0],rand[1]),max(rand[0],rand[1])))).replace("[A]",str(min(rand[0],rand[1]))).replace("[B]",str(max(rand[0],rand[1])))) |
| 21 |  |
| 22 | except ValueError: *# Выполняет нижеуказанный код в случае, если ошибка была найдена* |
| 23 | bot.send\_message(chat.id, setTag(data[lang]["error"]["numberRandomError"], None)) |

Итог

На этом документация подходит к концу. Надеемся, что вы нашли в ней то, что искали. Если же у вас остались вопросы, постарайтесь найти ответы на них в интернете или в нейросетях. В крайних случаях можно обратиться в поддержку.

Берегите себя! By qpikzz.

Автор • Исходный код • Пользовательское соглашение