**Урок 7**

**Создание кнопок InlineKeyboardMarkup**

На предыдущем уроке мы создали кнопки, которые отображались вместо основной клавиатуры на экране. Научились создавать меню. Познакомились с библиотекой **types**, классами **ReplyKeyboardMarkup** и **KeyboardButton**

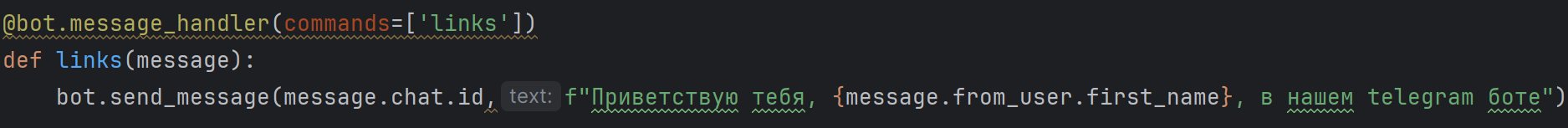
Сегодня мы создадим inline-кнопки, связанные с сообщениями в чате. При этом пользователь видит и основную клавиатуру. Создаются они с помощью метода **InlineKeyboardMarkup**. С помощью таких кнопок мы запрограммируем бота так, чтобы он давал пользователю полезные ссылки.

**Самостоятельная работа**

Добавьте боту новую команду **/** **links.**

**Встроенные кнопки**

Ранее мы делали так, чтобы при получении команды бот отправлял текст пользователю «Приветствую тебя, {имя пользователя}, в нашем telegram боте». Давайте же теперь будем при получении команды /**links** будем отправлять нашему пользователю тот же самый текст, но только уже с кнопками **inline**.

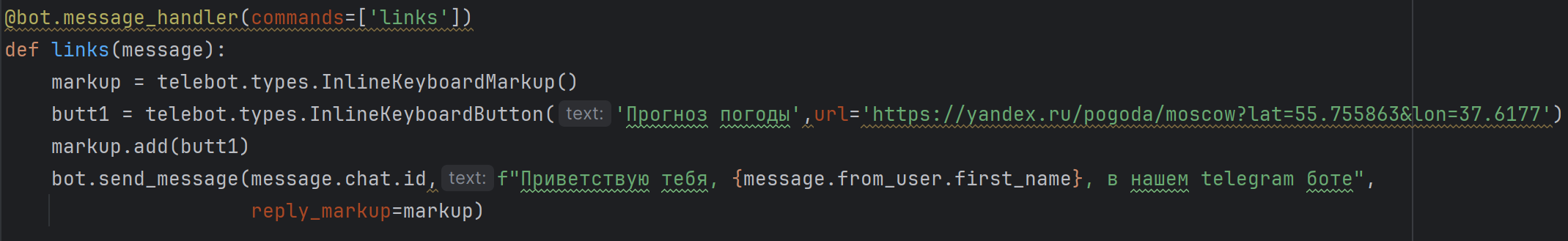


Возьмем нашу программу, написанную на предыдущих занятиях, поменяем команду и добавим в нашу функцию программу для появления кнопок.

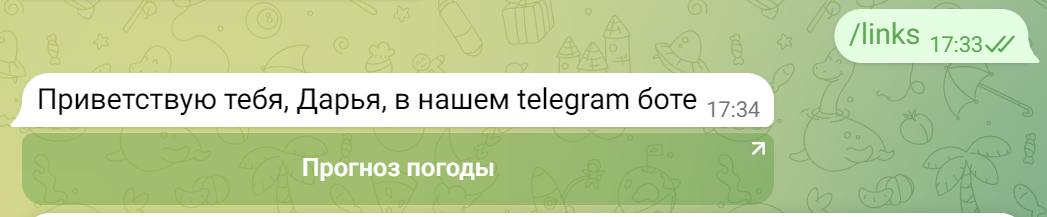
Для создания таких кнопок мы будем использовать точно такую же структуру, как мы это делали для reply-кнопок. Тут будет отличаться только то, что вместо **ReplyKeyboardMarkup()** мы будем использовать **InlineKeyboardMarkup()**, а также для самих кнопок нам надо использовать класс **InlineKeyboardButton()**

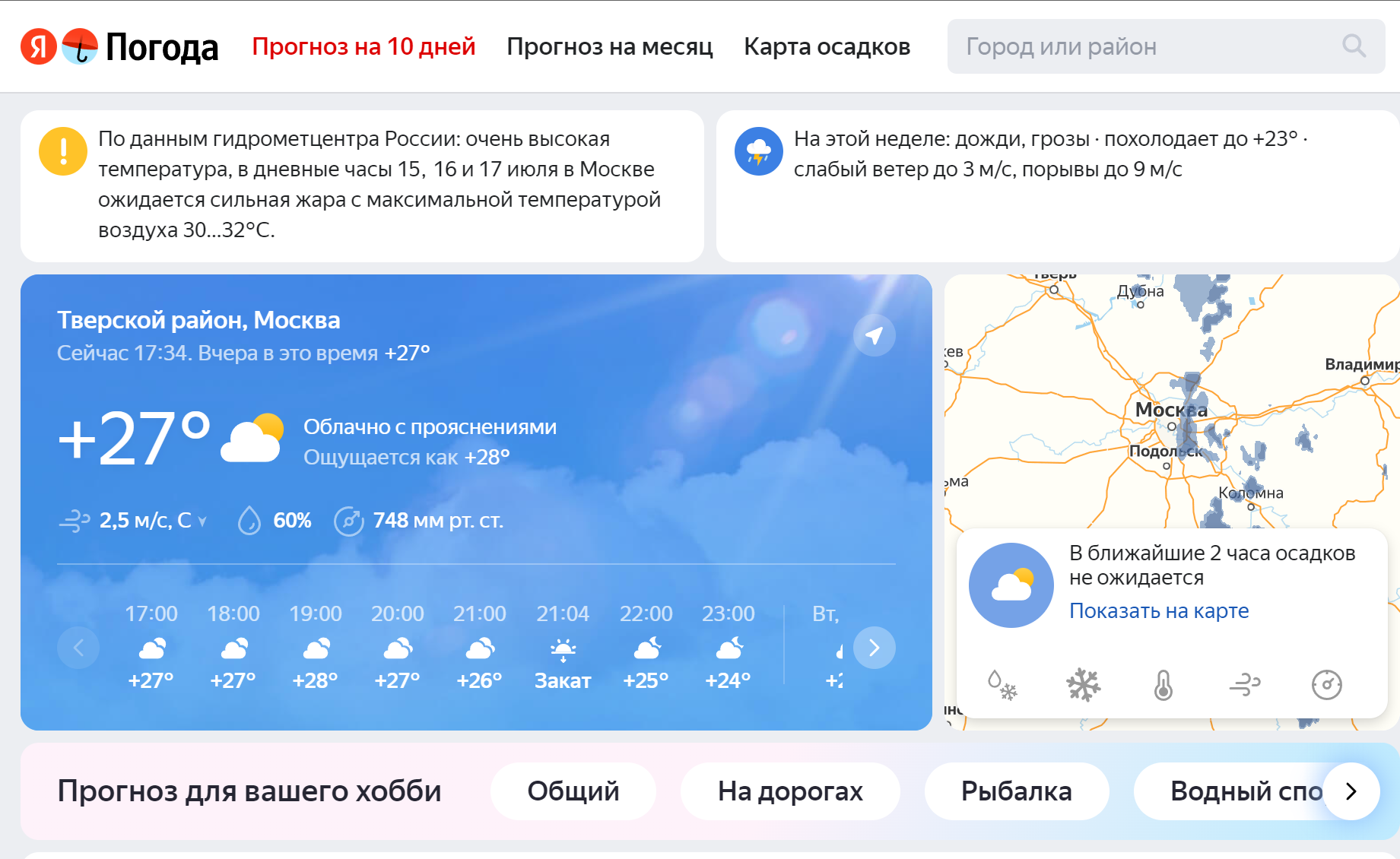
Создадим markup — объект, где будет храниться клавиатура. Далее мы создаём с вами встроенные кнопки, поэтому нас интересует класс **InlineKeyboardMarkup()**.

Чтобы добавить новую кнопку, мы обращаемся к **InlineKeyboardButton**. В качестве параметров мы должны указать текст для кнопки (у меня это будет «Прогноз погоды») и также, если мы переходим по какой-либо ссылке, нам надо в качестве параметра указать **url** и здесь же мы указываем саму ссылку, которая будет открываться при нажатии на данную кнопку (в моем случае будет открываться страница с погодой). Теперь выведем нашу кнопку с сообщением. В методе **send\_message** мы передаем параметр **reply\_markup** и в качестве значения передаем наш объект **markup**.



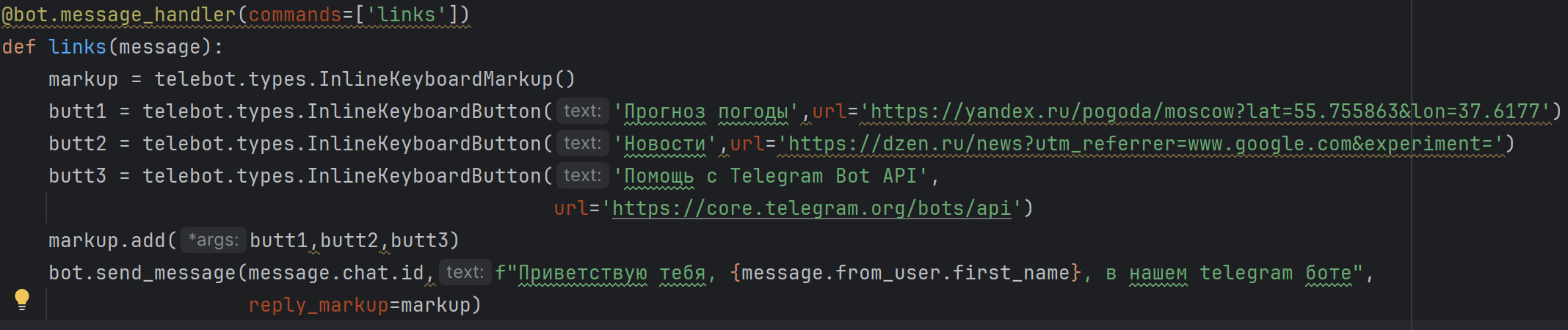
Теперь при запуске бота, когда мы введем команду /**links**, то наш бот выдаст нам текст, а также нашу встроенную кнопку, при нажатии на которую мы будем переходить на страницу с прогнозом погоды.

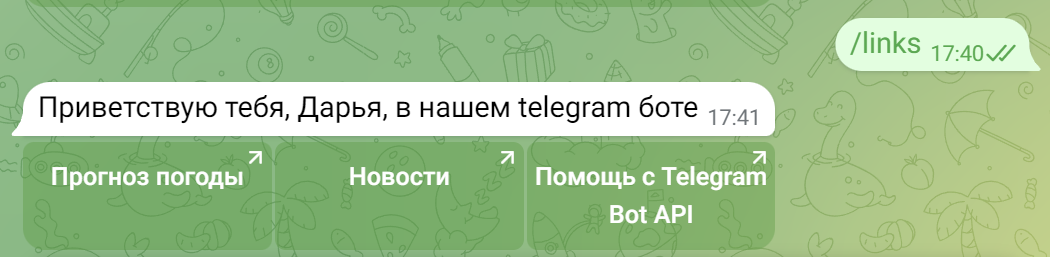




**Самостоятельная работа**

Добавьте несколько кнопок с разными полезными ссылками (например, новости и справочник по командам библиотеки telebot)



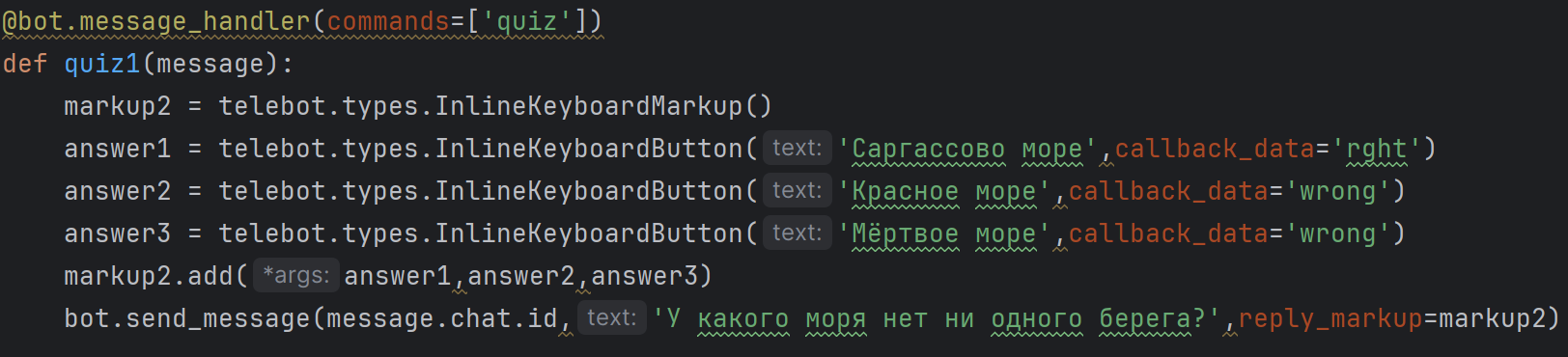


**Мини-викторина с ботом**

Давайте теперь добавим еще одну команду для нашего бота (например, **/quiz)** и сделаем викторину, в которой пользователь должен ответить на вопрос, нажимая на одну из inline-кнопок. После этого бот говорит пользователю правильно он ответил или нет.

Кнопки будут добавляться точно также, только теперь у нас не будет **url** адреса, а вместо этого у нас будет такой параметр, как **callback\_data**. Для кнопки с правильным ответом я напишу 'rght', для неправильных — 'wrong'

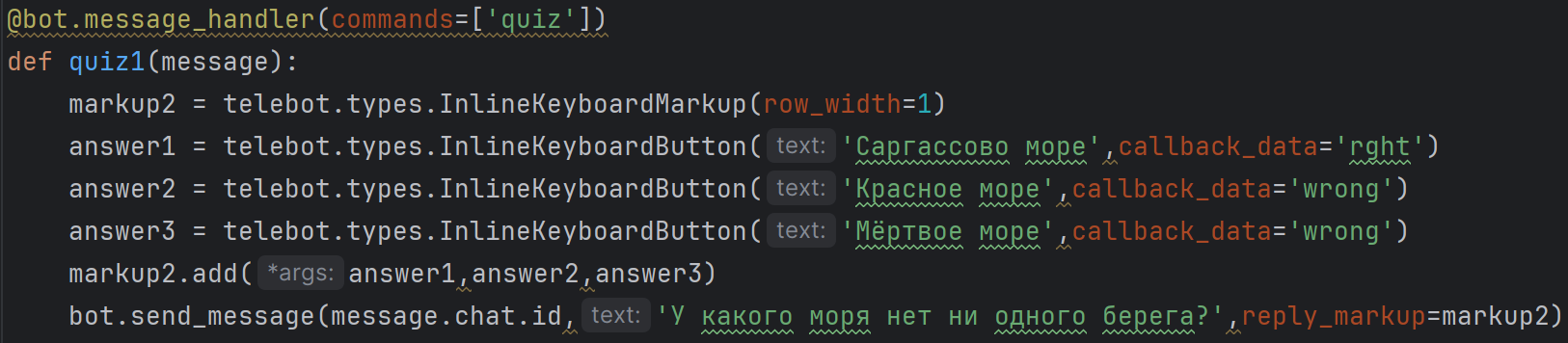
**callback\_data** означает то, что при нажатии на данную кнопку будет вызываться некая функция, которая как раз и будет отвечать за действия этой кнопки, и куда мы будем передавать некоторые значения.



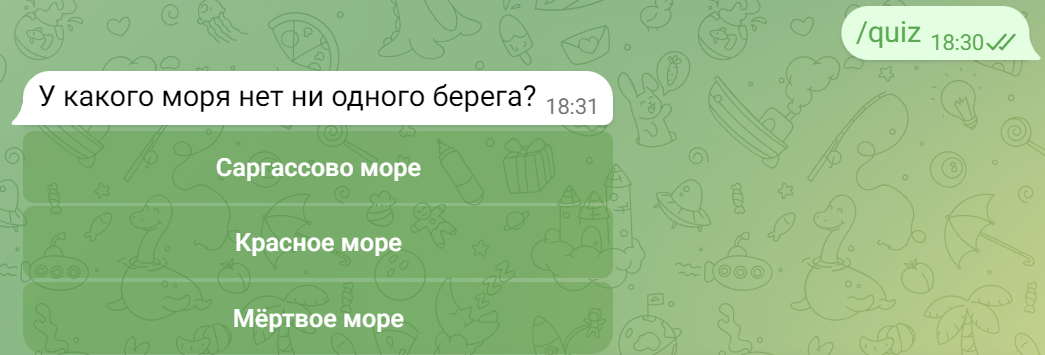
Теперь же если мы запустим бота, то у нас появятся наши кнопки, но они пока работать не будут.



Перед тем, как мы создадим функцию для обработки результата, мы разберёмся как можно располагать наши кнопки. Сейчас они находятся друг рядом с другом. Для чтобы поместить кнопки друг под другом, нам надо в команде **InlineKeyboardMarkup()** указать параметр **row\_width.** Этот параметр определяет количество кнопок в одном ряду. Мы укажем значение 1, чтобы на одной строке располагалась одна кнопка.



Посмотрим на результат:

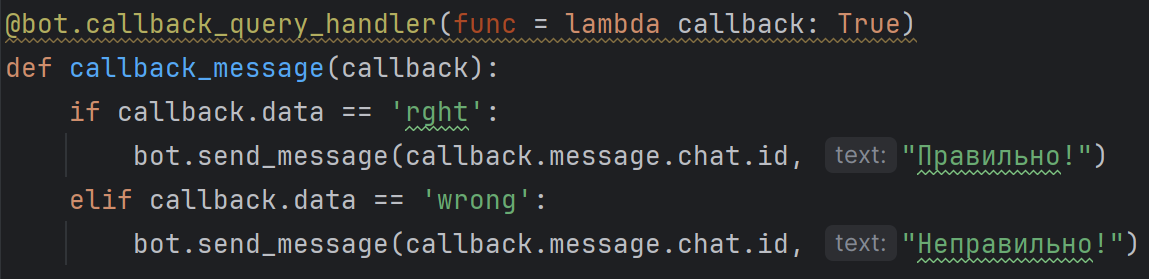


Давайте теперь напишем функцию, которая будет приводить в работу наши кнопки. Для этого мы должны прописать **@bot.callback\_query\_handler() —** это специальный декоратор для обработки параметров **callback\_data**. Внутри декоратора мы пишем анонимную функцию, где мы говорим один неким параметр, в случае если он будет пустым, то возвращаем значение True (func = lambda callback: True).

Далее мы создаем функцию, которая принимает один параметр — **callback**.

Теперь обращаемся к **callback.data** и получаем то, что передается при нажатии на некую кнопку. Например, в случае если пользователь нажимает на кнопку «Саргассово море», то у нас отправляется **callback\_data** с такой информацией, как rght. Далее необходимо проверить какой ответ выбрал пользователь (на какую кнопку он нажал) и какой **callback\_data** получила наша функция. Если получила rght, бот напишет «Правильно!», а если wrong — напишет «Неправильно!»

Для этого прописываем **bot.send\_message** с которым мы ознакомились в прошлых уроках. А в качестве параметра — callback.message.chat.



Протестируем работу бота:



**Самостоятельная работа**

Добавьте в викторину несколько вопросов с вариантами ответов в виде inline-кнопок.

**Рефлексия**

* Сегодня мы создали inline-кнопки, связанные с сообщениями в чате
* Познакомились с методом **InlineKeyboardMarkup**
* С помощью кнопок мы запрограммировали бота так, чтобы он давал пользователю полезные ссылки
* Создали мини-викторину с ботом
* Познакомились с параметром callback\_data

***Тайминг.***

|  |  |  |
| --- | --- | --- |
|  | Раздел урока | Время с начала урока, мин. |
|  | Повторение материала предыдущего урока. Постановка цели урока | 0 |
|  | Самостоятельная работа | 10 |
|  | Встроенные кнопки | 15 |
|  | Самостоятельная работа | 35 |
|  | Мини-викторина с ботом | 45 |
|  | Бескомпьютерная деятельность | 65 |
|  | Самостоятельная работа | 70 |
|  | Рефлексия | 85 |