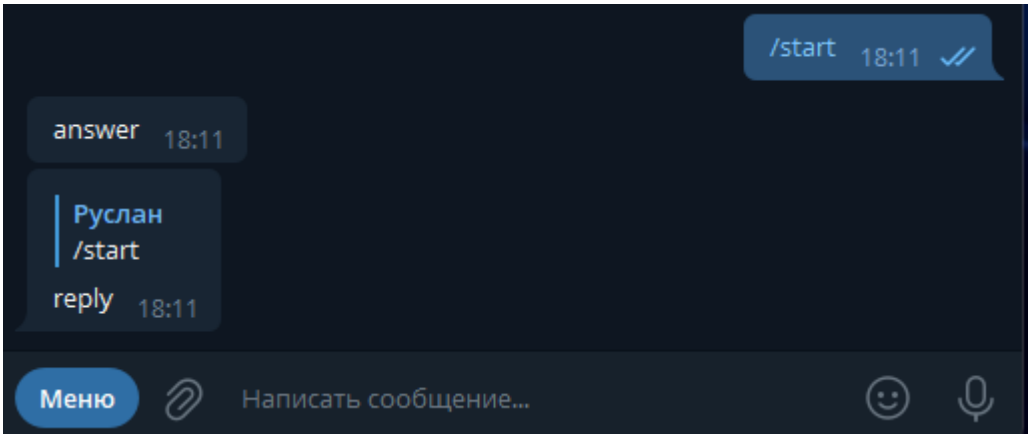
Теоретические материалы к занятию 1

Прошлые занятия мы с вами научились обрабатывать сообщения от пользователя и отправлять в ответ пользователю информацию. Единственное хотелось бы добавить разницу между message.answer и message.reply. Первое отправляет пользователю сообщение, второе отправляет пользователю сообщение с прикрепленным сообщением от пользователя.



Теперь уточнив данную особенность с чистой совестью перейдем к созданию клавиатур. Для начала разберем два существующих типа клавиатур.

ReplyKeyboardMarkup — это шаблоны сообщений. К примеру, ваш бот задает пользователю вопрос и предлагает варианты ответа. Пользователь может самостоятельно напечатать ответ, либо нажать на готовую кнопку. Такая клавиатура показывается вместо основной и не привязана ни к какому сообщению. В кнопки такой клавиатуры нельзя заложить никакой информации, нельзя запрограммировать для неё подобный если пользователь нажимает кнопку с текстом «abc» отправить текст «qwerty» алгоритм, отправлено будет только то, что написано на кнопке..

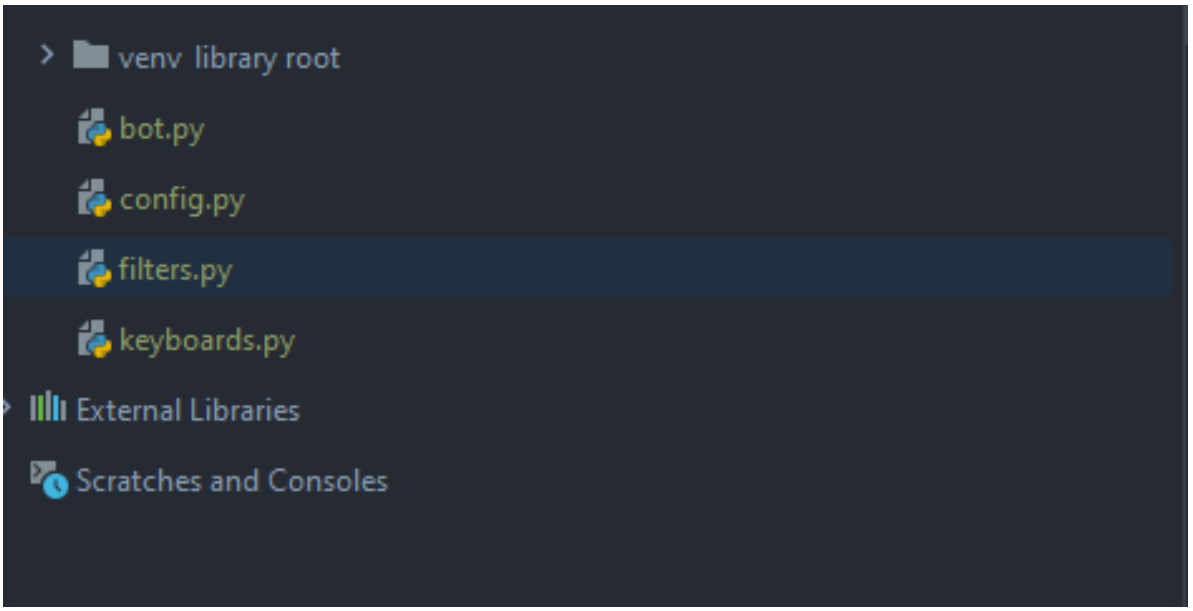
InlineKeyboardMarkup — это уже настоящая кастомная клавиатура. С её помощью мы можем выполнять более сложные действия. Она привязывается к сообщению, с которым была отправлена. В кнопки можно заложить любой текст размером от 1 до 64 байт. Инлайн кнопки позволяют скрыть в себе внутреннюю телеграм ссылку, ссылку на внешний ресурс, а также шорткат для инлайн запроса.

И ту и другую клавиатуру можно редактировать, но разными способами. Первая обновляется при отправке сообщения с новой клавиатурой типа ReplyKeyboardMarkup, вторую можно редактировать вместе с сообщением, к которому она прикреплена (или только саму разметку).

Начнем мы пожалуй с самой простой клавиатуры - ReplyKeyboardMarkup. Создадим шаблон-заготовку нашего бота.

```
import logging

from aiogram import Bot, Dispatcher, executor, types
from config import BOT_TOKEN

# Объект бота
bot = Bot(token=BOT_TOKEN, parse_mode=types.ParseMode.HTML)
# Диспетчер для бота
dp = Dispatcher(bot)
# Включаем логирование, чтобы не пропустить важные сообщения
logging.basicConfig(level=logging.INFO)


if __name__ == "__main__":
    # Запуск бота
    executor.start_polling(dp, skip_updates=True)
```

Теперь давайте создадим отдельный файл keyboards.py, кстати хендлеры выносить в отдельный файл тоже желательно, но мы в качестве упрощения этого не делали.



В первую очередь импортируем необходимые нам классы и создаём первую клавиатуру

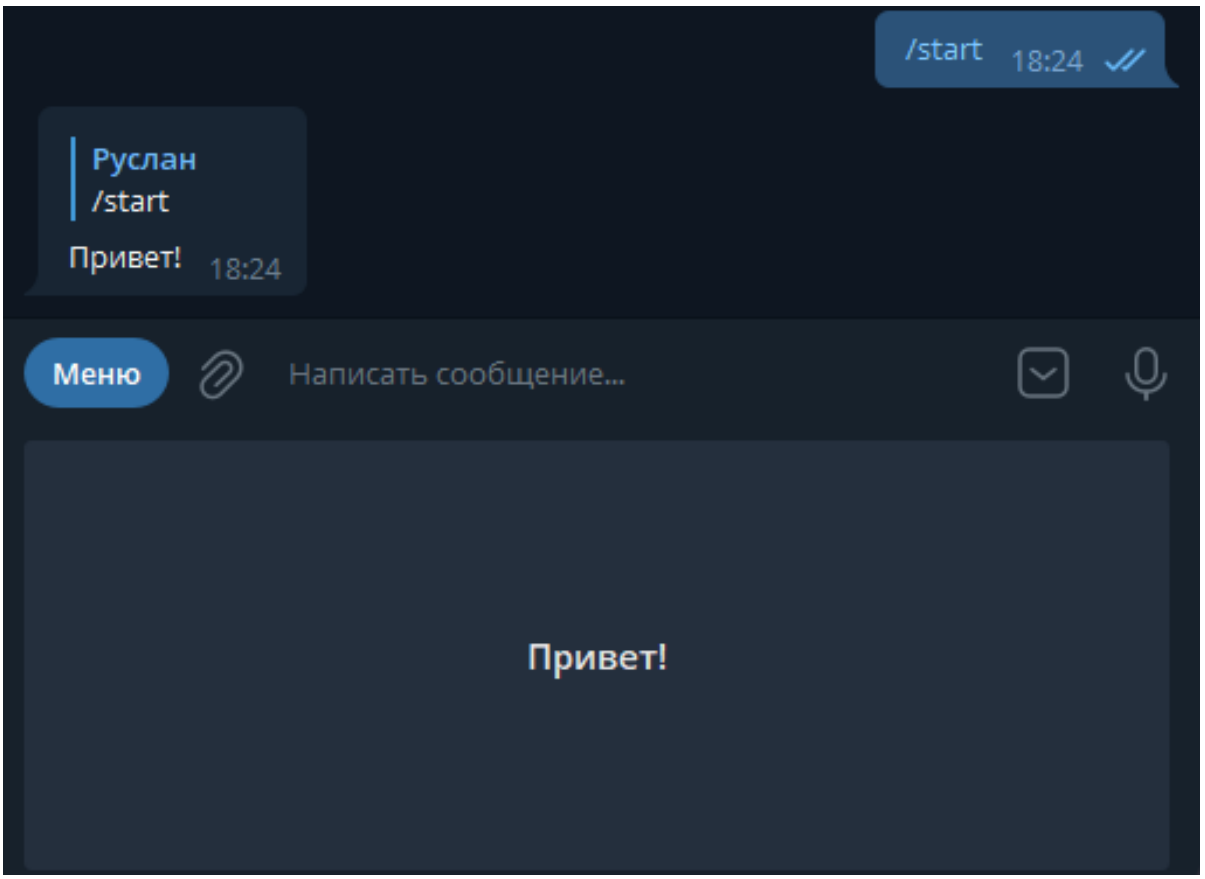
```
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton

button1 = KeyboardButton('Привет!')
```

```
kb = ReplyKeyboardMarkup()
kb.add(button1)
```

При инициализации класса KeyboardButton необходимо передать один обязательный параметр - текст, который указан на кнопке и который пользователь будет отправлять по нажатию на эту кнопку. У объекта класса ReplyKeyboardMarkup есть несколько методов, позволяющих добавить кнопку, в данном случае мы используем add. Таким образом мы получили первую клавиатуру.

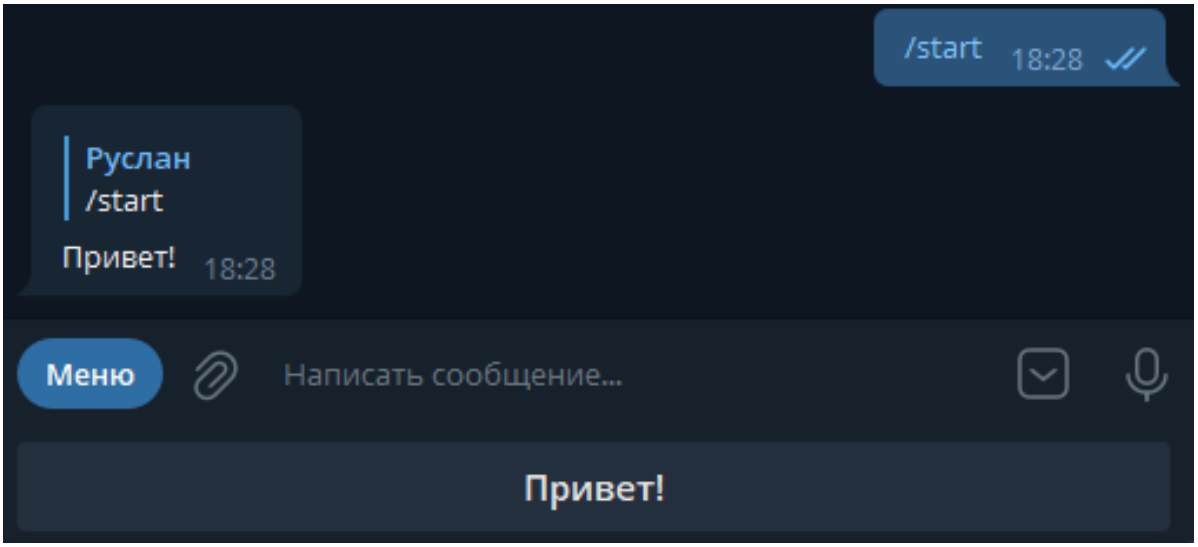
Создаём обработчик, который будет отправлять нам сообщение и наш шаблон (отправить отдельно клавиатуру никак нельзя, она является параметром к сообщению).

```
import logging

from aiogram import Bot, Dispatcher, executor, types
from config import BOT_TOKEN
from keyboards import kb

# Объект бота
bot = Bot(token=BOT_TOKEN, parse_mode=types.ParseMode.HTML)
# Диспетчер для бота
dp = Dispatcher(bot)
# Включаем логирование, чтобы не пропустить важные сообщения
logging.basicConfig(level=logging.INFO)


@dp.message_handler(commands=['start'])
async def process_start_command(message: types.Message):
    await message.reply("Привет!", reply_markup=kb)


if __name__ == "__main__":
    # Запуск бота
    executor.start_polling(dp, skip_updates=True)
```

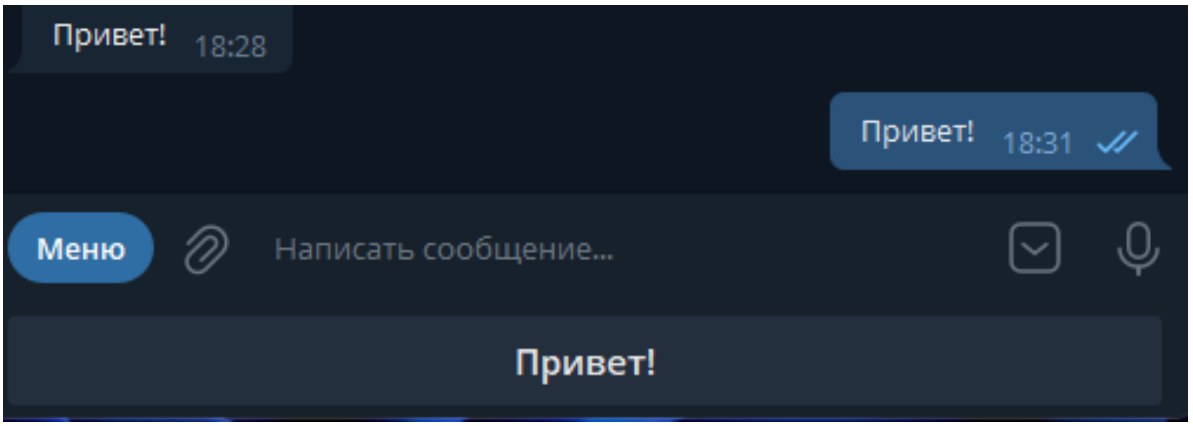


У нас появилась клавиатура и одна кнопка.Эта кнопка с маленьким текстом занимает очень много места. Телеграм позволяет автоматически уменьшить размер, для этого необходимо передать в инициализатор класса ReplyKeyboardMarkup параметру resize_keyboard значение True. Изменим нашу клавиатуру:

```
kb = ReplyKeyboardMarkup(resize_keyboard=True)
kb.add(button1)
```



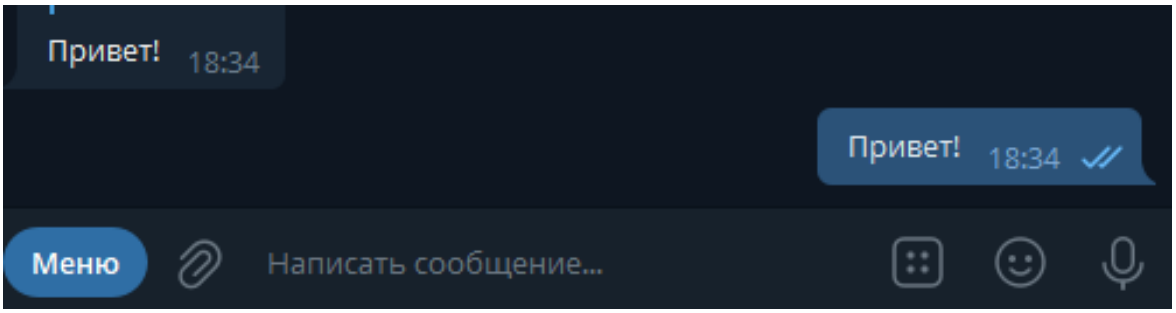
При нажатии на кнопку, мы отправляем в качестве сообщения текст указанный на ней.



Все бы хорошо, но после нажатия клавиатура у нас никуда не пропадает. Давайте опять изменим нашу клавиатуру, добавив новый параметр

```
kb = ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)
kb.add(button1)
```

Теперь при нажатии на кнопку, клавиатура автоматический скрывается



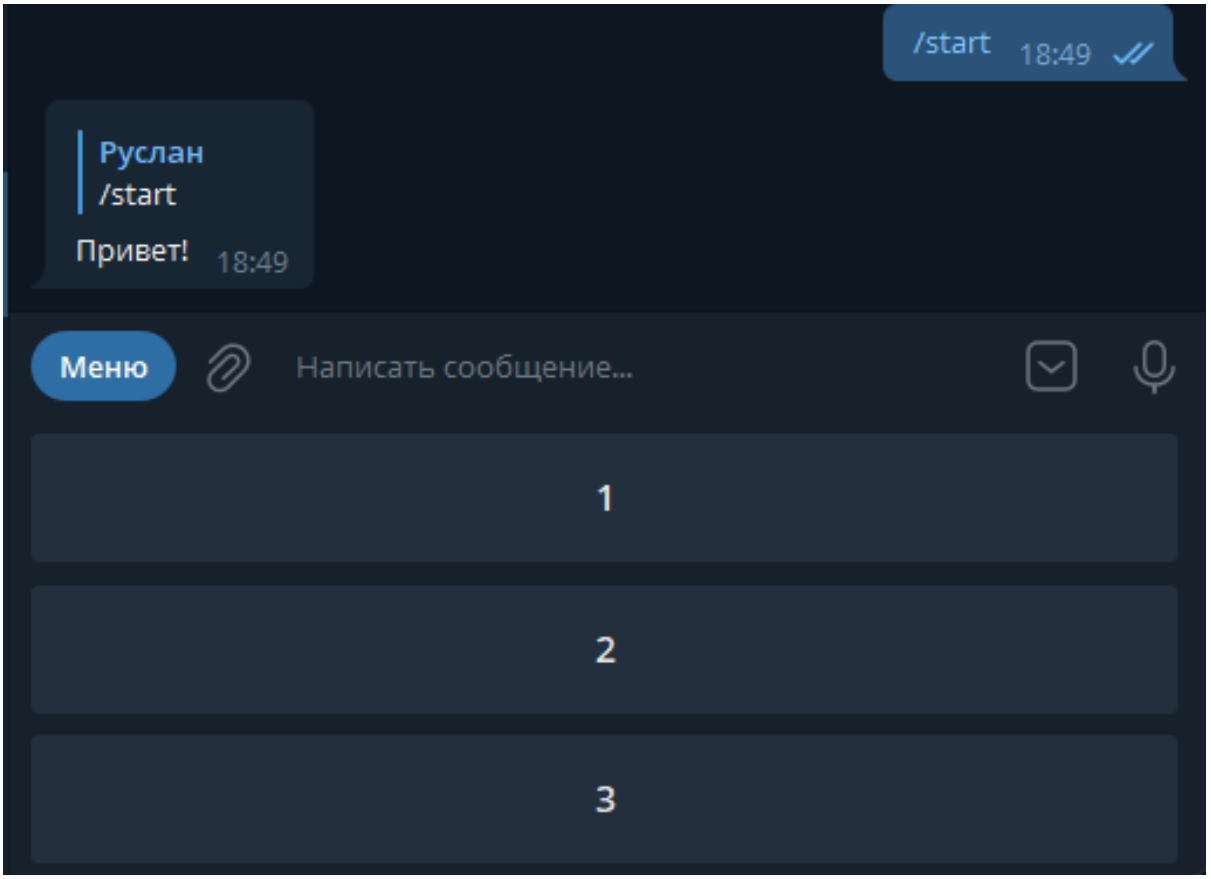
Конечно же для реализации на подобной клавиатуре бота, так как при нажатии на кнопку уходит ее текст, нам придется прописывать соответствующие обработчики.

Рассмотрим подробно работу встроенных методов для создания более сложных клавиатур, а именно row, insert и add. Создаём кнопки, которые мы сможем использовать повторно и генерируем несколько разных клавиатур:

```
button1 = KeyboardButton('1')
button2 = KeyboardButton('2')
button3 = KeyboardButton('3')

kb1 = ReplyKeyboardMarkup().add(
```

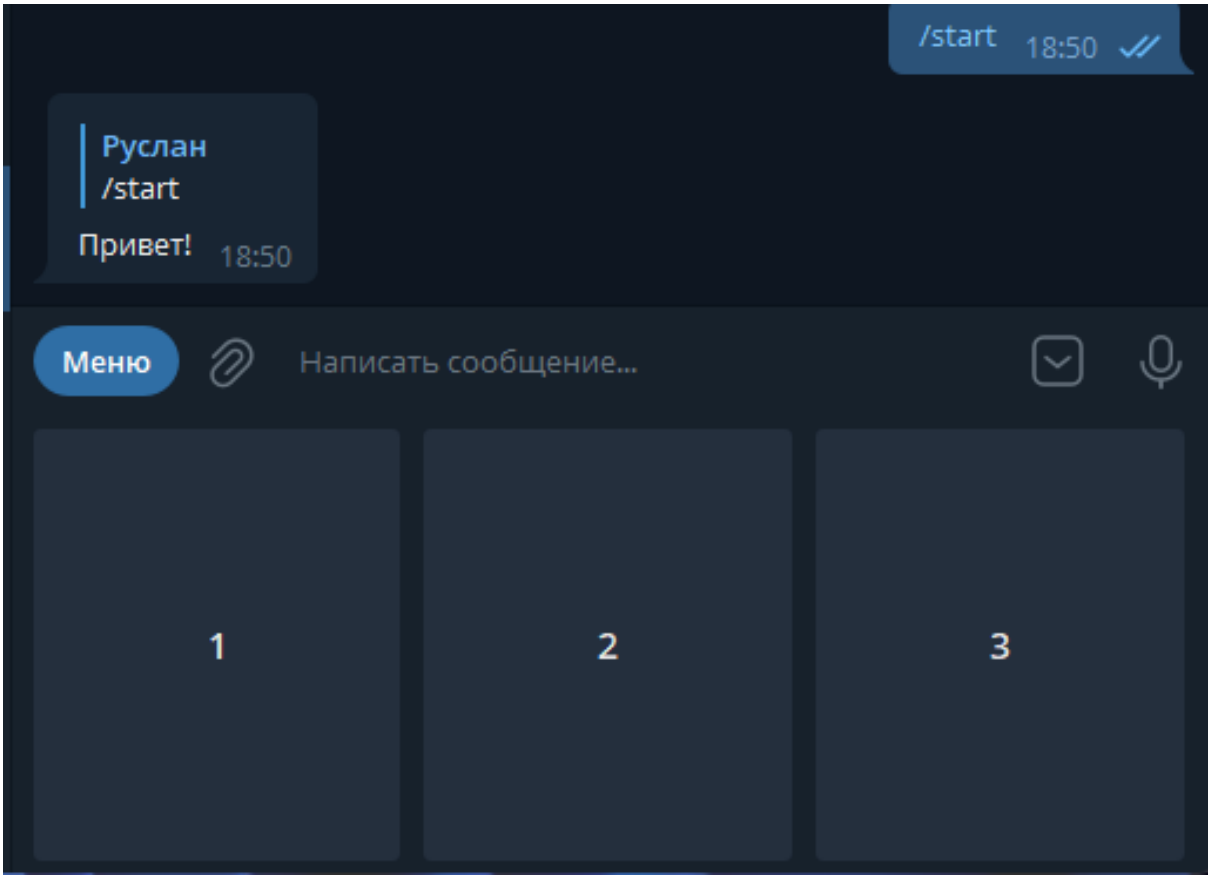
```
   button1).add(button2).add(button3)
```

```
@dp.message_handler(commands=['start'])
async def process_start_command(message: types.Message):
   await message.reply("Привет!", reply_markup=kb1)
```



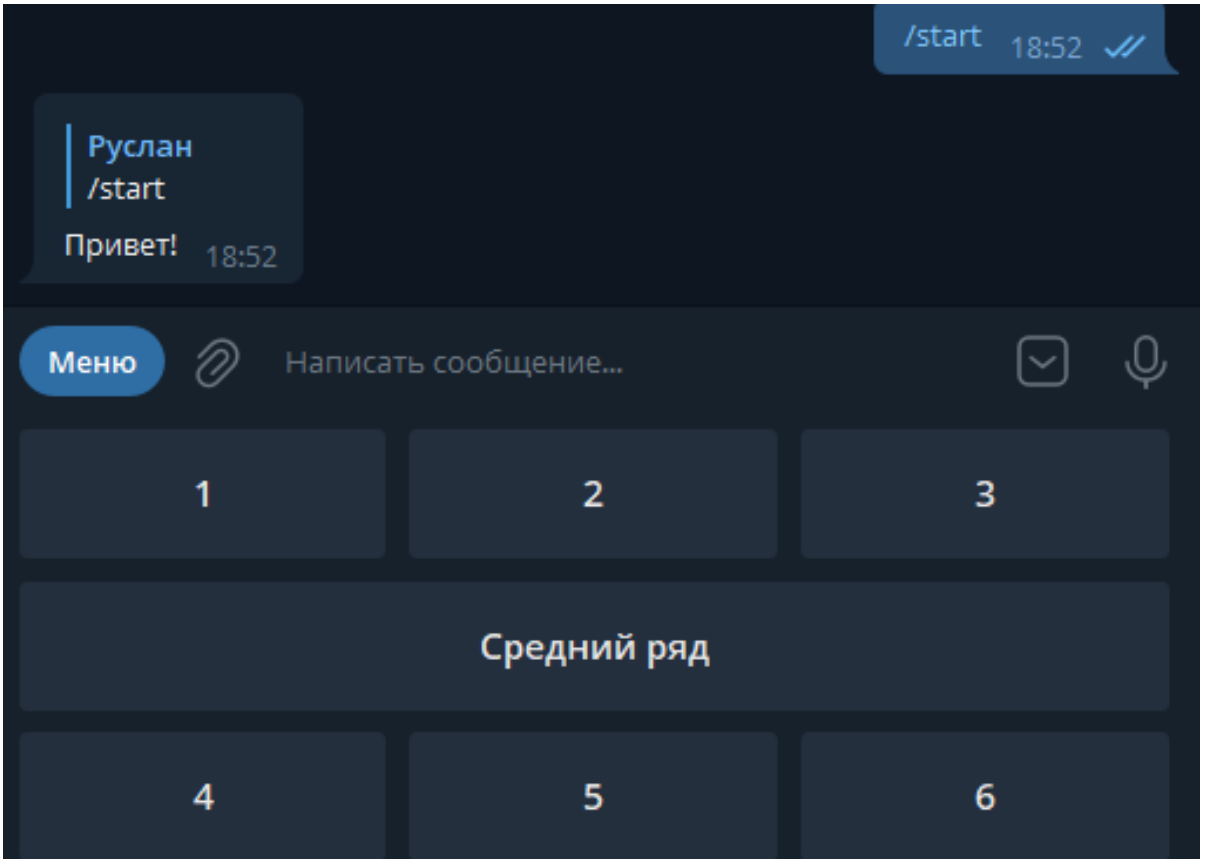

```
kb2 = ReplyKeyboardMarkup().row(
   button1, button2, button3
)
```

```
@dp.message_handler(commands=['start'])
async def process_start_command(message: types.Message):
   await message.reply("Привет!", reply_markup=kb2)
```




```
kb3 = ReplyKeyboardMarkup().row(
    button1, button2, button3
).add(KeyboardButton('Средний ряд'))

button4 = KeyboardButton('4')
button5 = KeyboardButton('5')
button6 = KeyboardButton('6')
kb3.row(button4, button5)
kb3.insert(button6)
```

```
@dp.message_handler(commands=['start'])
async def process_start_command(message: types.Message):
    await message.reply("Привет!", reply_markup=kb3)
```



- Метод add принимает в себя любое количество кнопок, всегда начинает добавление с новой строки и переносит ряд при достижении значения установленной ширины.
- Метод row тоже принимает любое количество, но при этом не переносит кнопки на новый ряд, а добавляет всё полученное в одну строчку.
- Метод insert работает по схеме схожей с add, но только начинает добавлять к последнему ряду. И только если там уже достигнута максимальная ширина, начинает новую строку.

ReplyKeyboardMarkup позволяет запросить у пользователя его контакт или локацию. В данном случае при нажатии кнопки будет отправлено не то, что написано на ней. Их можно отправлять как по одной, так и в составе более сложной клавиатуры. Добавим обе кнопки:

```
kb4 = ReplyKeyboardMarkup(resize_keyboard=True).add(
   KeyboardButton('Отправить свой контакт', request_contact=True)
).add(
   KeyboardButton('Отправить свою локацию',
request_location=True)
)
```

```
@dp.message_handler(commands=['start'])
async def process_start_command(message: types.Message):
    await message.reply("Привет!", reply_markup=kb4)
```