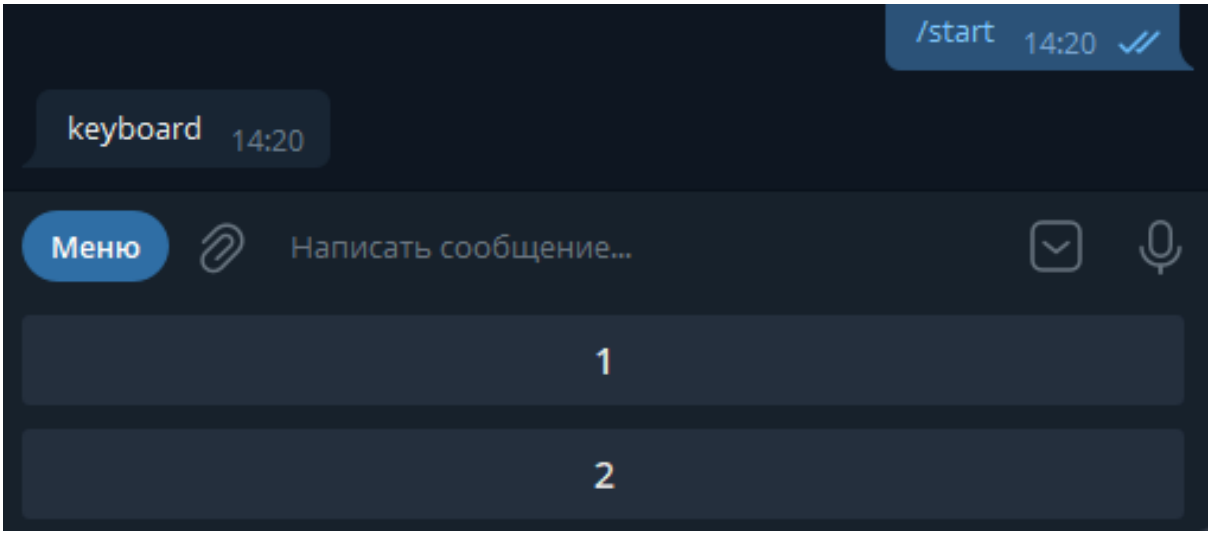
Теоретические материалы к занятию 4

Сегодня мы разовьем тему клавиатур, а вернее меню которые из них можно делать. Зная что обычные клавиатуры возвращают нам текст указанный на нажатой кнопке, а инлайн клавиатуры позволяют передавать значения в data, соответственно все это мы можем обрабатывать и перехватывать. Начнем с обычной клавиатуры

```
def get_simple_kb():
    kb = ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)
    btn1 = KeyboardButton(text='1')
    btn2 = KeyboardButton(text='2')
    kb.add(btn1)
    kb.add(btn2)
    return kb
```

```
@dp.message_handler(commands='start')
async def menu_bot(message: types.Message):
    await message.answer('keyboard', reply_markup=get_simple_kb())
```

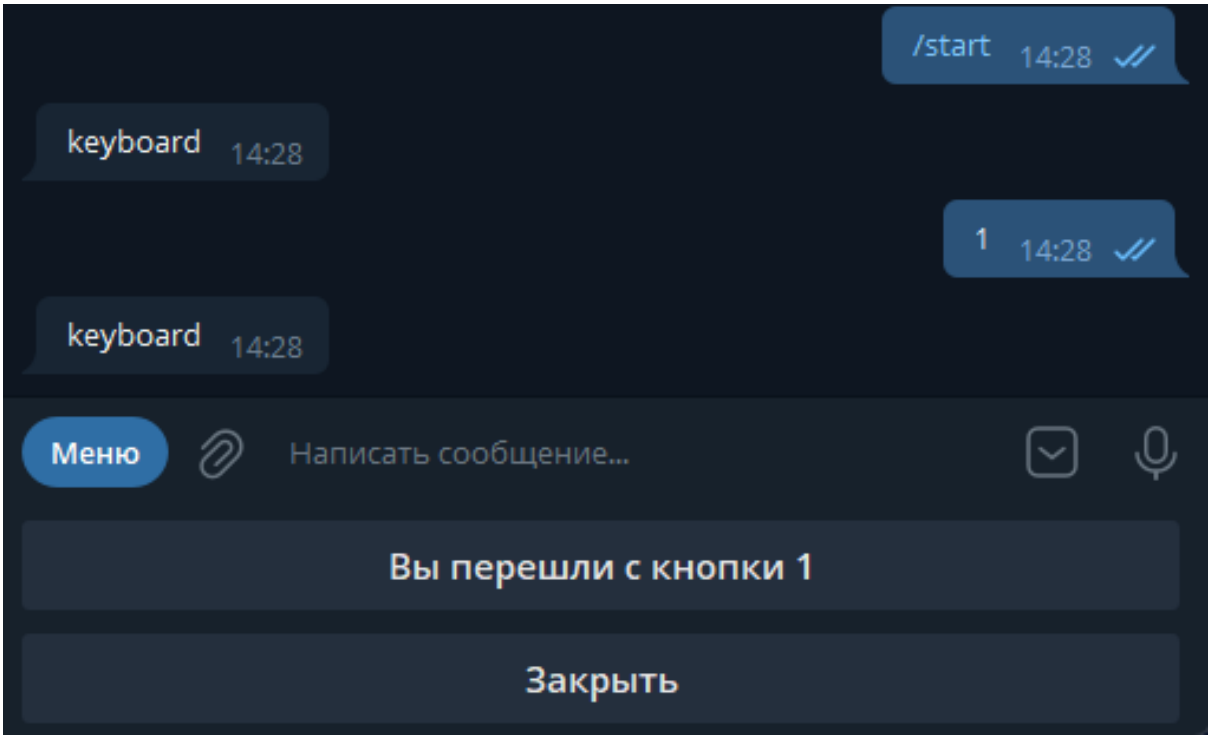


Теперь обработаем данные нажатия клавиш, в ответ выведем другое меню

```
def get_simple_kb1():
```

```
   kb = ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True).add(
        KeyboardButton('Вы перешли с кнопки
1')).add(KeyboardButton('Закрыть'))
   return kb
```

```
@dp.message_handler(Text(equals='1'))
async def menu_bot(message: types.Message):
   await message.answer('keyboard',
reply_markup=get_simple_kb1())
```



Сделаем тоже самое со второй кнопкой

```
def get_simple_kb2():
   kb = ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True).add(
        KeyboardButton('Вы перешли с кнопки
2')).add(KeyboardButton('Закрыть'))
   return kb
```

```
@dp.message_handler(Text(equals='2'))
async def menu_bot(message: types.Message):
   await message.answer('keyboard',
reply_markup=get_simple_kb2())
```

Мы можем продолжать сколько угодно увеличивать и расширять нашу клавиатуру. Давайте добавим на кнопку закрыть действие.

```
@dp.message_handler(Text(equals='Закрыть'))
async def menu_bot(message: types.Message):
   await message.answer('keyboard closed',
reply_markup=ReplyKeyboardRemove())
```

Попробуйте потренируйтесь и создайте многоуровневую простую клавиатуру. А теперь давайте подумаем как нам сделать тоже самое с инлайн клавиатурой. Обработав callback_data!

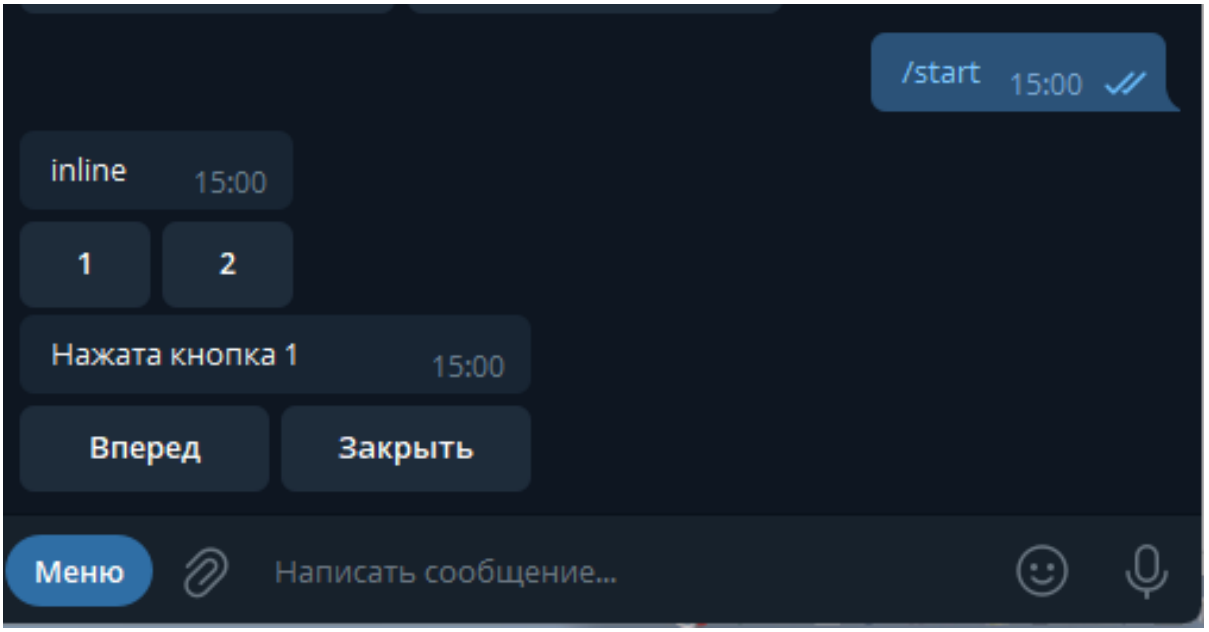
```
def get_inline_kb():
   inline_kb = InlineKeyboardMarkup()
   btns = [InlineKeyboardButton('1', callback_data='1'),
InlineKeyboardButton('2', callback_data='2')]
   inline_kb.add(*btns)
   return inline_kb
```

```
@dp.message_handler(commands='start')
async def menu_bot(message: types.Message):
   await message.answer('inline', reply_markup=get_inline_kb())
```

```
def get_inline_kb1():
   inline_kb = InlineKeyboardMarkup()
   btns = [InlineKeyboardButton('Вперед', callback_data='f1'),
           InlineKeyboardButton('Закрыть', callback_data='back')]
   inline_kb.add(*btns)
   return inline_kb
```

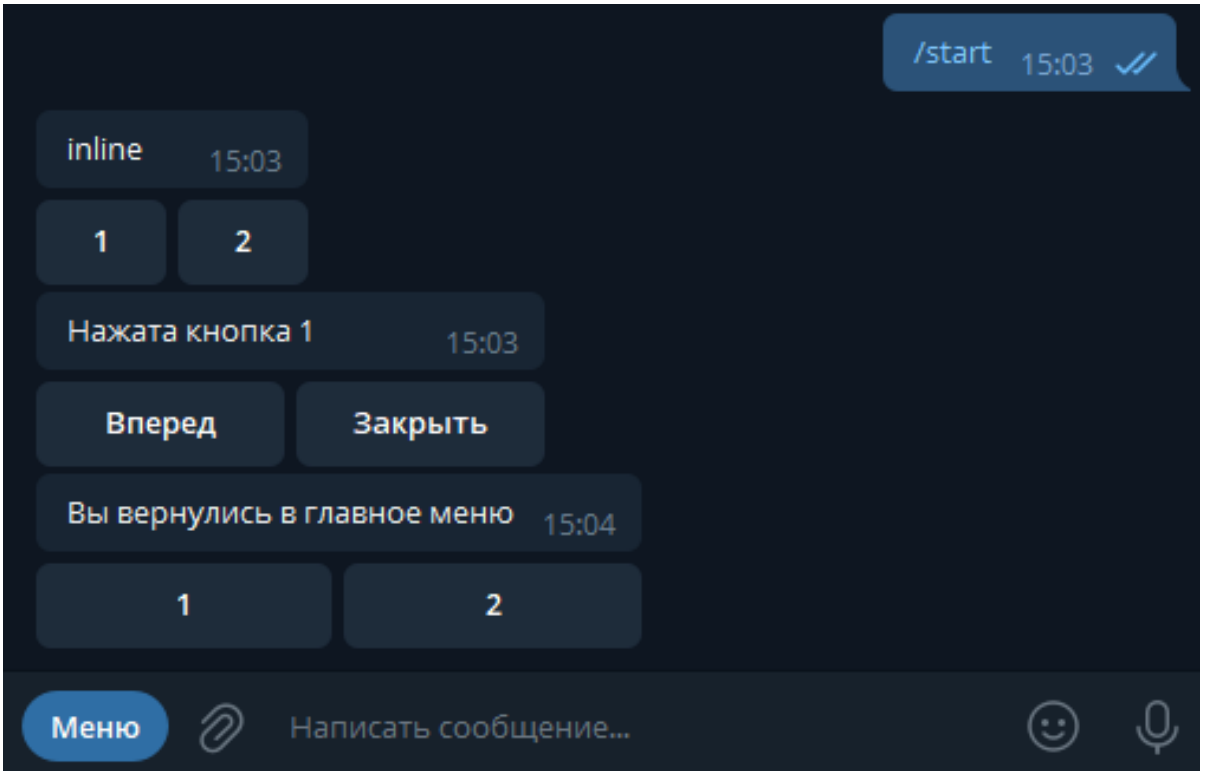
```
@dp.callback_query_handler(Text(equals='1'))
async def btn1(cb: types.CallbackQuery):
   await cb.answer()
```

```
   await cb.message.answer('Нажата кнопка 1',
reply_markup=get_inline_kb1())
```



Сделаем возврат после нажатия на кнопку закрыть на основное меню

```
@dp.callback_query_handler(Text(equals='back'))
async def btn1(cb: types.CallbackQuery):
   await cb.answer()
   await cb.message.answer('Вы вернулись в главное меню',
reply_markup=get_inline_kb())
```



Давайте немного расслабимся и напишем простую игру угадай число для нашего бота. Бот будет загадывать число, а мы будем пытаться угадать его. Давайте начнем с шаблона бота

```
from aiogram import Bot, Dispatcher, executor, types


BOT_TOKEN = os.getenv('TOKEN')
bot = Bot(token=BOT_TOKEN)
dp = Dispatcher(bot)


if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)
```

использовать сложные меню из клавиатуры я думаю мы тут не будем и максимально упростим логику.
Создадим функцию, которая возвращает секретное число от 1 до 10

```
def random_num():
    secret_number = random.randint(1, 10)
    return secret_number
```

Создадим первый обработчик для команды /start

```
@dp.message_handler(commands=['start'])
async def start_command(message:types.Message):
    await message.answer('Привет! Это игра угадай число!\nДля
старта игры нажми на кнопку начать')
```

Добавим стартовую клавиатуру и добавим ее в обработчик

```
start_menu = InlineKeyboardMarkup(resize_keyboard=True)
button_start = InlineKeyboardButton('Начать',
callback_data='startgame')
button_cancel = InlineKeyboardButton('Отмена',
callback_data='cancel')
start_menu.add(button_start, button_cancel)
```

```
@dp.message_handler(commands=['start'])
async def start_command(message:types.Message):
    await message.answer('Привет! Это игра угадай число!\nДля
старта игры нажми на кнопку начать', reply_markup=start_menu)
```

Так как у нас есть функция возвращающая секретное число, а также мы реализуем систему попыток, то сразу добавим переменные

```
secret_number = None
attempts = None
```

Теперь давайте напишем обработчики для наших callback кнопок

```
@dp.callback_query_handler(Text(equals='cancel'))
async def end_game_command(call: types.CallbackQuery):
    await call.message.delete()
```

Данный обработчик у нас будет просто закрывать все удаляя сообщение. В обработчике начала игры же мы установим случайное число, кол-во попыток, а также сообщение о начале игры

```
@dp.callback_query_handler(Text(equals='startgame'))
```

```
async def start_game_command(call:types.CallbackQuery):
    global secret_number, attempts
    attempts = 3
    secret_number = random_num()
    await call.message.answer('Я загадал число от 1 до
10\nПопробуй угадай его')
    await bot.answer_callback_query(call.id)
    await call.message.delete()
    print(secret_number)
```

Секретное число мы выводим в консоль, только для того чтобы свериться!

Теперь необходимо обрабатывать число введенное пользователем, для этого мы напишем соответствующий обработчик.

```
dp.message_handler(regexp=re.compile(r'^\d+$'))
async def check_numbers(message:types.Message):
   global secret_number
   user_number = int(message.text)
   if user_number > secret_number:
       await message.reply(f'Секретное число меньше.\nКоличество
попыток: {attempts}')
   elif user_number < secret_number:
       await message.reply(f'Секретное число больше.\nКоличество
попыток: {attempts}')
   else:
       await message.reply('Вы угадали!\nНачать
заново?',reply_markup=start_menu)
       secret_number = None
```

Так отлично! Но может произойти ситуация, что секретное число у нас останется None, да и попытки надо бы уменьшать. Давайте модифицируем наш код

```
dp.message_handler(regexp=re.compile(r'^\d+$'))
async def check_numbers(message:types.Message):
   global secret_number, attempts
   user_number = int(message.text)
   # print(user_number)
   attempts -= 1
   if secret_number != None:
       if user_number > secret_number:
           await message.reply(f'Секретное число
меньше.\nКоличество попыток: {attempts}')
       elif user_number < secret_number:
```

```
            await message.reply(f'Секретное число
больше.\nКоличество попыток: {attempts}')
        else:
            await message.reply('Вы угадали!\nНачать
заново?',reply_markup=start_menu)
            secret_number = None
            attempts = None
```

Ну и напоследок добавим условия проигрыша

```
@dp.message_handler(regexp=re.compile(r'^\d+$'))
async def check_numbers(message:types.Message):
    global secret_number, attempts
    user_number = int(message.text)
    # print(user_number)
    attempts -= 1
    if secret_number != None:
        if user_number > secret_number:
            await message.reply(f'Секретное число
меньше.\nКоличество попыток: {attempts}')
        elif user_number < secret_number:
            await message.reply(f'Секретное число
больше.\nКоличество попыток: {attempts}')
        else:
            await message.reply('Вы угадали!\nНачать
заново?',reply_markup=start_menu)
            secret_number = None
            attempts = None


    else:
        await message.reply('Для начала игры введите /start')
    if attempts < 1:
        await message.reply('Вы проиграли',
reply_markup=start_menu)
```

Наша игра готова.