



# Специалист по системам искусственного интеллекта Модуль 7.3

МГТУ им. Н.Э. Баумана





# Часть 1 (Создание базы данных пользователей)



# Регистрация нового пользователя

- Будем хранить данные пользователей в отдельном файле, к которому будем обращаться по необходимости. Для этого добавим 2 функции

```
def load_user_data():  
    if os.path.exists('user_data.pkl'):  
        with open('user_data.pkl', 'rb') as f:  
            return pickle.load(f)  
    return {}
```

```
def save_user_data(data):  
    with open('user_data.pkl', 'wb') as f:  
        pickle.dump(data, f)
```

# Регистрация нового пользователя

Также создадим функцию для проверки, что пользователь уже зарегистрирован.

```
def is_user_registered(user_id):  
    user_data = load_user_data()  
    print(user_data)  
    return str(user_id) in user_data and user_data[str(user_id)].get('registered', False)
```

# Регистрация нового пользователя

Когда пользователь нажимает кнопку “start” нам необходимо проверить, что он уже зарегистрирован в системе. Модифицируем функцию.

```
if is_user_registered(user_id):  
    # Пользователь уже зарегистрирован  
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)  
    btn2 = types.KeyboardButton("Меню")  
    markup.add(btn2)  
    send_message = f'<b>Привет {message.from_user.first_name}</b>\nНажми на кнопку ниже, чтобы  
начать'  
    bot.send_message(message.chat.id, send_message, parse_mode='html', reply_markup=markup)  
else:  
    # Пользователь не зарегистрирован  
    Register_menu(message)
```

# Регистрация нового пользователя

Также необходимо создать меню регистрации пользователя. Для этого создадим новую команду register.

```
@bot.message_handler(commands=['register'])
def Register_menu(message):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)
    # Добавляем кнопку для запроса номера телефона
    btn_phone = types.KeyboardButton("Отправить телефон", request_contact=True)
    markup.add(btn_phone)

    bot.send_message(message.chat.id,
        "Просьба пройти регистрацию, чтобы использовать все возможности бота\n\n"
        "1. Нажмите 'Отправить телефон'\n"
        "2. Затем введите ваше ФИО",
        reply_markup=markup)
```

# Регистрация нового пользователя

При нажатии кнопки “Отправить телефон” телеграмм отправит в бот информацию контакта, которую необходимо обработать.



```
@bot.message_handler(content_types=['contact'])
def handle_contact(message):
    user_id = str(message.from_user.id)
    phone = message.contact.phone_number
    user_data = load_user_data()
    if user_id not in user_data:
        user_data[user_id] = {}
    user_data[user_id]['phone'] = phone
    save_user_data(user_data)
    msg = bot.send_message(message.chat.id, "Теперь введите ваше ФИО:")
    bot.register_next_step_handler(msg, process_name_step)
```

# Регистрация нового пользователя

Затем бот направит запрос на ввод ФИО пользователя. Добавим функцию, которая запустится в результате вызова “register\_next\_step\_handler”.

```
def process_name_step(message):  
    user_id = str(message.from_user.id)  
    full_name = message.text  
    user_data = load_user_data()  
    if user_id not in user_data:  
        user_data[user_id] = {}  
    user_data[user_id]['full_name'] = full_name  
    user_data[user_id]['registered'] = True  
    save_user_data(user_data)  
  
    bot.send_message(message.chat.id, "✅ Регистрация завершена! Теперь вам доступны все функции бота.")  
    Main_menu(message)
```





## Часть 2 (Последовательное изучение материалов каждым учеником)

# Последовательное изучение материалов

Каждый учащийся проходит материалы в своём темпе. Необходимо реализовать индивидуальное отображение курсов для каждого пользователя. Для этого добавим ещё полей в файл.

```
user_data[user_id]['full_name'] = full_name
user_data[user_id]['registered'] = True
user_data[user_id]['middle_score'] = 0
user_data[user_id]['level'] = 1
user_data[user_id]['level_pass'] = False
```

# Последовательное изучение материалов

Создадим список курсов с наименованием и ссылками на обучающие материалы.

```
lessons = {  
    'Урок 1: Изучение сложения' : 'https://music.yandex.ru/' ,  
    'Урок 2: Изучение умножения' : 'https://music.yandex.ru/' ,  
    'Урок 3: Изучение деления' : 'https://music.yandex.ru/' ,  
    'Урок 4: Изучение квадрата' : 'https://music.yandex.ru/' ,  
    'Урок 4: Изучение взятие корня' : 'https://music.yandex.ru/' ,  
}
```

# Последовательное изучение материалов

Добавим новую кнопку в меню и напомним алгоритм автоматического показа нужных кнопок в зависимости от того сколько курсов уже прошёл пользователь.

```
elif message.text == 'Начать обучение':  
    Main_menu(message)  
    level = load_user_data()[str(message.from_user.id)]['level']  
    markup_line = types.InlineKeyboardMarkup()  
    for i, lesson in enumerate(lessons.keys()):  
        if i <= level-1:  
            btn = types.InlineKeyboardButton(text=lesson, url=lessons[lesson])  
            markup_line.add(btn)  
    bot.send_message(message.from_user.id, "Для обучения тебе доступны материалы. Пройди материал, чтобы открыть доступ к следующему. Всего 5 уроков!",  
                      reply_markup=markup_line)
```



## Часть 3 (Тестирование ученика по материалам)



# Создание тест системы для учеников

Чтобы открывать доступ ученику к следующим материалам необходимо провести его тестирование. Добавим кнопку и начнём прописывать логику

```
elif message.text == 'Пройти тестирование по модулю': # Обработка начала тестирования
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    level = load_user_data()[str(message.from_user.id)][ 'level' ] # Получение уровня
    for i in range(level): # Создание кнопок для доступных тестов
        markup.add(str(i+1))
    markup.add(main_button)
    msg = bot.send_message( # Запрос выбора модуля
        message.chat.id,
        f'Выберите модуль по которому хотите пройти тестирование',
        reply_markup=markup
    )
    bot.register_next_step_handler(msg, test_mode) # Регистрация обработчика выбора
```

# Создание тест системы для учеников

Создадим txt файл с вопросами. Добавим туда структуру - Вопрос\_ответ 1\_ответ 2\_ответ 3\_ответ 4\_номер правильного ответа

Сколько будет 2+2? 1 2 3 4 3

Сколько будет 3+3? 1 3 4 6 3

Сколько будет 1+10? 11 21 12 32 0

Также добавим несколько глобальный переменных

`user_progress = {}` # Словарь для хранения прогресса пользователей в тестах

`questions = []` # Список для хранения вопросов теста

# Создание тест системы для учеников

Добавим функцию, которая вызывается при выборе пользователем доступного теста

```
def test_mode(message):
    global questions # Использование глобальной переменной questions
    global user_progress # Использование глобальной переменной user_progress
    if message.text == "Меню": # Если пользователь ввел "Меню"
        Main_menu(message) # Вызов главного меню
        return
    user_progress[message.from_user.id] = [0,0,''] # Инициализация прогресса пользователя
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2) # Создание клавиатуры
    markup.add(main_button) # Добавление основной кнопки
    bot.send_message(message.from_user.id, f'Начинаем тестирование по теме -
{list(lessons.keys())[int(message.text)-1]}', reply_markup=markup) # Сообщение о начале теста
    try:
        questions = open('test_'+str(message.text)+'.txt').readlines() # Чтение вопросов из файла
    except BaseException:
        bot.send_message(message.from_user.id, f'Ой. Что-то сломалось, попробуйте в другой раз.') #
        Сообщение об ошибке
        Main_menu(message) # Возврат в главное меню
        return
    print(questions) # Вывод вопросов в консоль (для отладки)
    send_question(message, questions[user_progress[message.from_user.id][0]]) # Отправка первого
    вопроса
```



# Создание тест системы для учеников

Добавим функцию, которая будет генерировать сообщение с вопросом и кнопками для выбора ответа.

```
# Функция отправки вопроса пользователю
def send_question(message, question):
    markup = types.InlineKeyboardMarkup(row_width=2) # Создание inline-клавиатуры
    for i, answer in enumerate(question.split('_')[1:-1]): # Разбиение строки вопроса на части
        btn = types.InlineKeyboardButton( # Создание кнопки для каждого варианта ответа
            text=answer,

callback_data=f'answer_{user_progress[message.from_user.id][0]}_{i}_{question.split("_")[-1]}' #
Формат callback_data: answer_номер_вопроса_номер_ответа_правильный_ответ
        )
        markup.add(btn) # Добавление кнопки в клавиатуру
    msg = bot.send_message( # Отправка сообщения с вопросом
        message.from_user.id,
        f'{user_progress[message.from_user.id][0]+1}. Вопрос: {question.split("_")[0]}', # Текст
        reply_markup=markup # Прикрепление клавиатуры
    )
    user_progress[message.from_user.id][2] = msg.message_id # Сохранение ID сообщения
```

# Создание тест системы для учеников

Добавим обработчик события, когда пользователь отправляет нам ответ на вопрос,

```
# Обработчик ответов на вопросы теста
@bot.callback_query_handler(func=lambda call: call.data.startswith("answer_"))
def handle_answer(message):
    _, ques, answ, corr = message.data.split('_') # Разбор callback_data
    bot.edit_message_text( # Редактирование сообщения с вопросом
        chat_id=message.from_user.id,
        message_id=user_progress[message.from_user.id][2],
        text=questions[int(ques)].split('_')[0]+"Номер ответа - "+str(int(answ)+1), # Новый
        текст сообщения
        reply_markup=None # Удаление кнопок
    )
    user_progress[message.from_user.id][0] += 1 # Увеличение счетчика вопросов
    if int(answ) == int(corr): # Проверка правильности ответа
        user_progress[message.from_user.id][1] += 1 # Увеличение счетчика правильных ответов
    if user_progress[message.from_user.id][0] != len(questions): # Если вопросы не закончились
        bot.send_message(message.from_user.id, "Следующий вопрос") # Уведомление о следующем
        вопросе
        send_question(message, questions[user_progress[message.from_user.id][0]]) # Отправка
        следующего вопроса
```

# Создание тест системы для учеников

Добавим обработчик события, когда пользователь отправляет нам ответ на вопрос,

```
else: # Если вопросы закончились
    score = round(user_progress[message.from_user.id][1]*100/len(questions),2) # Расчет
результата
    bot.send_message(message.from_user.id, "Тест завершён") # Сообщение о завершении
    bot.send_message(message.from_user.id, f"Ты правильно ответил на -
{user_progress[message.from_user.id][1]} из {len(questions)} вопросов") # Статистика
    bot.send_message(message.from_user.id, f"Твой результат за первый модуль {score}%") #
Результат в процентах
    user_data = load_user_data() # Загрузка данных пользователей
    user_id = str(message.from_user.id)
    user_data[user_id]['middle_score'] += score # Обновление среднего балла
    save_user_data(user_data) # Сохранение данных
    if score >=50: # Если проходной балл достигнут
        bot.send_message(message.from_user.id, f"Ты успешно прошёл тестирование по уровню
{user_data[user_id]['level']}. Теперь можешь приступить к следующему")
        user_data[user_id]['level'] += 1 # Повышение уровня
        save_user_data(user_data) # Сохранение данных
    else: # Если проходной балл не достигнут
        bot.send_message(message.from_user.id, f"Ты не набрал проходной минимум, попробуй
снова.")
```

# Создание тест системы для учеников

Добавим кнопку для отображения статистики ученика,

```
elif message.text == 'Мой текущий балл': # Обработка запроса баллов
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)
    markup.add(main_button)
    score = load_user_data()[str(message.from_user.id)][ 'middle_score' ] #
    Получение баллов
    bot.send_message(message.chat.id, f'Ваш текущий средний балл: {score}') #
    Отправка баллов
```



## Часть 4 (Админ-меню)