Московский государственный технический

университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «ПиКЯП»

Отчет по лабораторной работе №5 и №6

«Разработка бота на основе конечного автомата для Telegram с использованием языка Python»

Выполнил:	Проверил:
студент группы ИУ5-36Б	преподаватель каф. ИУ5
Рухлин Алексей	Нардид А. Н.

Москва, 2024 г.

Описание задания

Задание:

1. Разработайте простого бота для Telegram. Бот должен использовать функциональность создания кнопок.
2. Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

Текст программы

```
import telebot

import random

from telebot import types



bot =
telebot.TeleBot('7455958243:AAHMGsQtBT9BW21H4Pf-7RVpMK
2KNgz3H6I');



# Замените на ваш токен

API_TOKEN =
'7455958243:AAHMGsQtBT9BW21H4Pf-7RVpMK2KNgz3H6I'



player = {

  "hp": 3,

  "level": 1,
```

```
  "xp": 0,
}


player_hp = 3


enemies = [
  {"name": "Слизь", "hp": random.randint(1, 6),
"image_url": "https://i.imgur.com/v5cBBwD.png"},
  {"name": "Гоблин", "hp": random.randint(1, 6),
"image_url": "https://i.imgur.com/2KUVTk6.png"},
  {"name": "Троль", "hp": random.randint(1, 6),
"image_url": "https://i.imgur.com/yLz1VXn.png"}
]


def roll_dice():
  return random.randint(1, 6)


def attack(player_attack, enemy_defense):
  return player_attack > enemy_defense


def battle(enemy, message):
  player_turn = True


  def send_battle_options():
```

```
      markup = types.InlineKeyboardMarkup(row_width=2)

      attack_button =
types.InlineKeyboardButton("Атаковать",
callback_data="attack")

      flee_button =
types.InlineKeyboardButton("Сбежать",
callback_data="flee")

      status_button =
types.InlineKeyboardButton("Статус",
callback_data="status")

      markup.add(attack_button, flee_button,
status_button)

      bot.send_message(message.chat.id, f"Ты
сражаешься с {enemy['name']}! У {enemy['name']}
осталось {enemy['hp']} HP.", reply_markup=markup)


  send_battle_options()


  @bot.callback_query_handler(func=lambda call: True)

  def handle_battle_action(call):

      nonlocal player_turn

      action = call.data


      if action == "attack":

          if player_turn:

              player_roll = roll_dice()

              enemy_roll = roll_dice()
```

```
            bot.edit_message_text(
                f"Ты атаковал! Бросок кубика:
{player_roll}. Враг защищается! Бросок кубика:
{enemy_roll}.",
                message.chat.id,
call.message.message_id)
            if attack(player_roll, enemy_roll):
                enemy['hp'] -= 1
                bot.send_message(message.chat.id,
f"Ты попал! {enemy['name']} потерял 1 HP. У
{enemy['name']} осталось {enemy['hp']} HP.")
            else:
                bot.send_message(message.chat.id,
f"Твой удар не попал! {enemy['name']} уклонился.")
            player_turn = False
        else:
            bot.send_message(message.chat.id,
"Сейчас ход врага!")


    elif action == "flee":
        if player_turn:
            flee_roll = roll_dice()
            bot.edit_message_text(f"Ты пытаешься
сбежать! Бросок кубика: {flee_roll}.",
message.chat.id,

call.message.message_id)
```

```
            if flee_roll >= 4:
                bot.send_message(message.chat.id,
f"Ты успешно сбежал от {enemy['name']}!")
                return
            else:
                bot.send_message(message.chat.id,
f"Попытка сбежать не удалась! {enemy['name']}
атакует!")
                player_turn = False
        else:
            bot.send_message(message.chat.id,
"Сейчас ход врага!")


    elif action == "status":
        bot.edit_message_text(
            f"Текущее здоровье: {player['hp']}
HP\nУровень: {player['level']}\nОпыт: {player['xp']}",
            message.chat.id,
call.message.message_id)



    if player['hp'] <= 0:
        bot.send_message(message.chat.id, "Ты был
повержен в бою... Игра окончена!")
        return
    if enemy['hp'] <= 0:
```

```
            bot.send_message(message.chat.id, f"Ты
победил {enemy['name']}!")

            return


        if not player_turn:

            enemy_roll = roll_dice()

            player_roll = roll_dice()

            bot.send_message(message.chat.id,

                             f"{enemy['name']} атакует!
Бросок кубика: {enemy_roll}. Ты защищаешься! Бросок
кубика: {player_roll}.")

            if attack(enemy_roll, player_roll):

                player['hp'] -= 1

                bot.send_message(message.chat.id,

                                 f"Ты получил удар! Ты
потерял 1 HP. Твоё здоровье: {player['hp']} HP.")

            else:

                bot.send_message(message.chat.id, f"Ты
уклонился от атаки {enemy['name']}!")

            player_turn = True


        send_battle_options()


@bot.message_handler(commands=['start'])

def send_welcome(message):
```

```
  player['chat_id'] = message.chat.id

  markup =
types.ReplyKeyboardMarkup(resize_keyboard=True)

  item1 = types.KeyboardButton("Начать приключение")

  item2 = types.KeyboardButton("Проверить статус")

  item3 = types.KeyboardButton("Пройти обучение")

  markup.add(item1, item2, item3)



  bot.send_message(message.chat.id, "Привет, искатель
приключений! Готов начать свое путешествие?",
reply_markup=markup)



@bot.message_handler(func=lambda message: message.text
== "Проверить статус")

def check_status(message):

  image_character = "https://i.imgur.com/xOtWfw1.png"

  bot.send_photo(message.chat.id, image_character)

  bot.send_message(message.chat.id, f"Ваше здоровье:
{player['hp']} HP\nУровень: {player['level']}\nОпыт:
{player['xp']}")



@bot.message_handler(func=lambda message: message.text
== "Пройти обучение")

def training(message):

  xp_gain = random.randint(10, 30)
```

```
  player['xp'] += xp_gain

  bot.send_message(message.chat.id, f"Вы прошли
обучение! Получили {xp_gain} опыта.")



  if player['xp'] >= player['level'] * 100:

      player['level'] += 1

      player['xp'] = 0

      player['hp'] += 1

      player_hp += 1

      bot.send_message(message.chat.id, f"Поздравляем!
Вы достигли нового уровня! Теперь ваш уровень:
{player['level']}. Ваше здоровье увеличилось на 1!
Текущее здоровье: {player['hp']}.")



  send_welcome(message)



@bot.message_handler(func=lambda message: message.text
== "Начать приключение")

def adventure(message):

  markup =
types.ReplyKeyboardMarkup(resize_keyboard=True)

  item1 = types.KeyboardButton("Исследовать")

  item2 = types.KeyboardButton("Отдыхать")

  item3 = types.KeyboardButton("Собирать ресурсы")

  item4 = types.KeyboardButton("Искать приключения")

  item5 = types.KeyboardButton("Назад")
```

```
  markup.add(item1, item2, item3, item4, item5)


  bot.send_message(message.chat.id, "Вы находитесь в
опасном мире, выберите действие:",
reply_markup=markup)


@bot.message_handler(func=lambda message: message.text
in ["Исследовать", "Искать приключения"])
def handle_action(message):
  action = message.text


  if action == "Исследовать":
      location = random.choice(["Лес", "Горы",
"Пещера", "Магическая поляна"])
      bot.send_message(message.chat.id, f"Вы
исследуете {location}. Вдруг появляется враг!
Готовьтесь к бою!")
      enemy = random.choice(enemies)
      bot.send_photo(message.chat.id,
enemy['image_url'])
      bot.send_message(message.chat.id, f"Вы встретили
врага: {enemy['name']} с {enemy['hp']} HP.")
      battle(enemy, message)
  elif action == "Искать приключения":
      enemy = random.choice(enemies)
      bot.send_message(message.chat.id, f"Вы встретили
врага: {enemy['name']} с {enemy['hp']} HP.")
```

```
        battle(enemy, message)


@bot.message_handler(func=lambda message: message.text
in ["Отдыхать", "Собирать ресурсы", "Назад"])
def handle_other_actions(message):
    action = message.text


    if action == "Отдыхать":
        image_action = "https://i.imgur.com/dDUeCVj.png"
        bot.send_photo(message.chat.id, image_action)


        bot.send_message(message.chat.id, "Вы решили
отдохнуть. Ваше здоровье восстанавливается на 1 HP.")
        player['hp'] = min(player['hp'] + 1, player_hp)
        bot.send_message(message.chat.id, f"Ваше текущее
здоровье: {player['hp']} HP.")
        send_adventure_options(message)


    elif action == "Собирать ресурсы":
        resources = random.choice(["дерево", "камни",
"цветы", "руды"])
        bot.send_message(message.chat.id, f"Вы собрали
{resources}.")
        bot.send_message(message.chat.id, f"Вы получили
опыт за собранные ресурсы! Опыт: {random.randint(5,
15)}.")
```

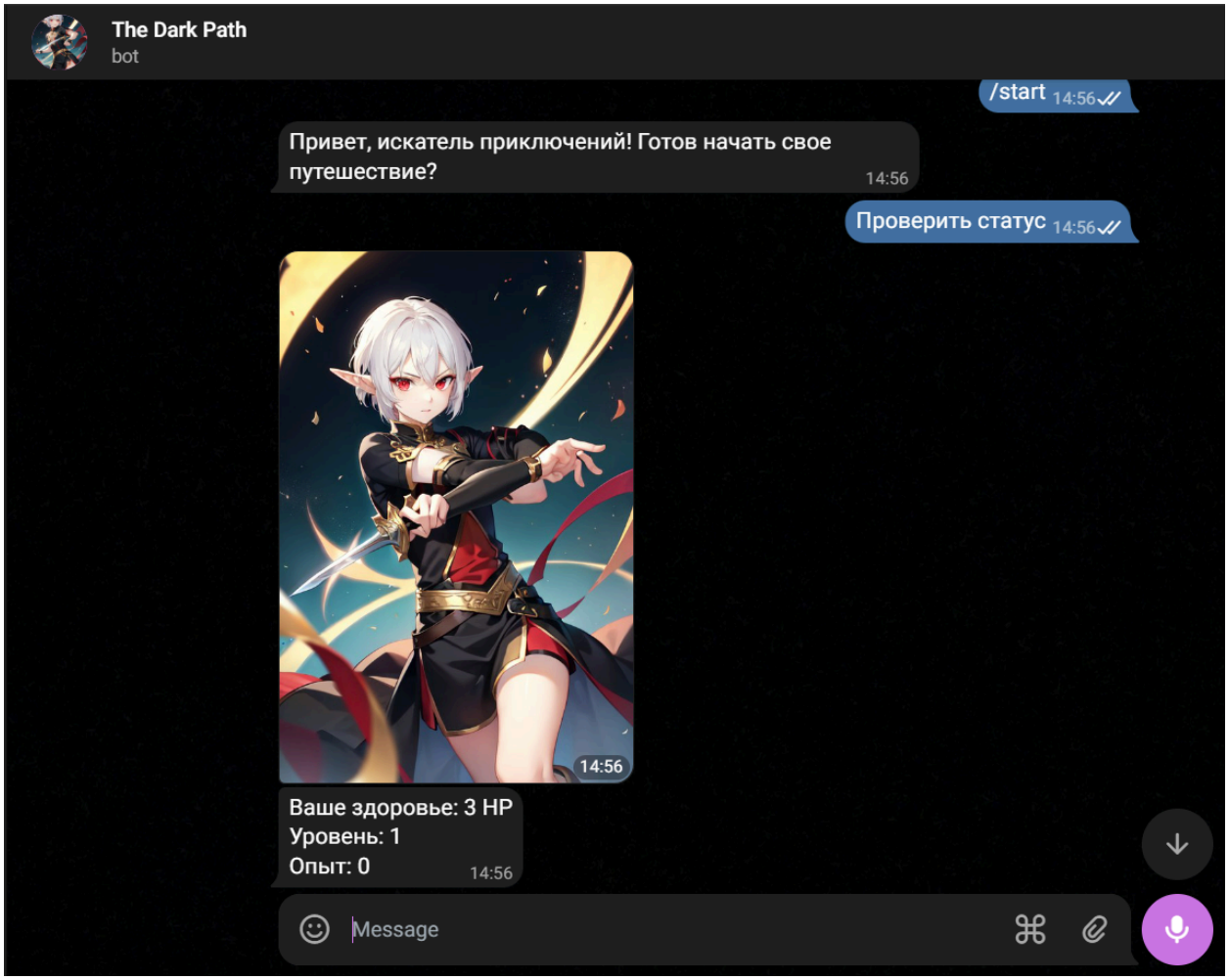
```
        player['xp'] += random.randint(5, 15)
        send_adventure_options(message)

    elif action == "Назад":
        send_welcome(message)

def send_adventure_options(message):
    markup =
types.ReplyKeyboardMarkup(resize_keyboard=True)
    item1 = types.KeyboardButton("Исследовать")
    item2 = types.KeyboardButton("Отдыхать")
    item3 = types.KeyboardButton("Собирать ресурсы")
    item4 = types.KeyboardButton("Искать приключения")
    item5 = types.KeyboardButton("Назад")
    markup.add(item1, item2, item3, item4, item5)

    bot.send_message(message.chat.id, "Вы находитесь в
опасном мире, выберите действие:",
reply_markup=markup)

@bot.message_handler(func=lambda message: message.text
== "Проверить статус")
def check_status(message):
    bot.send_message(message.chat.id,
```

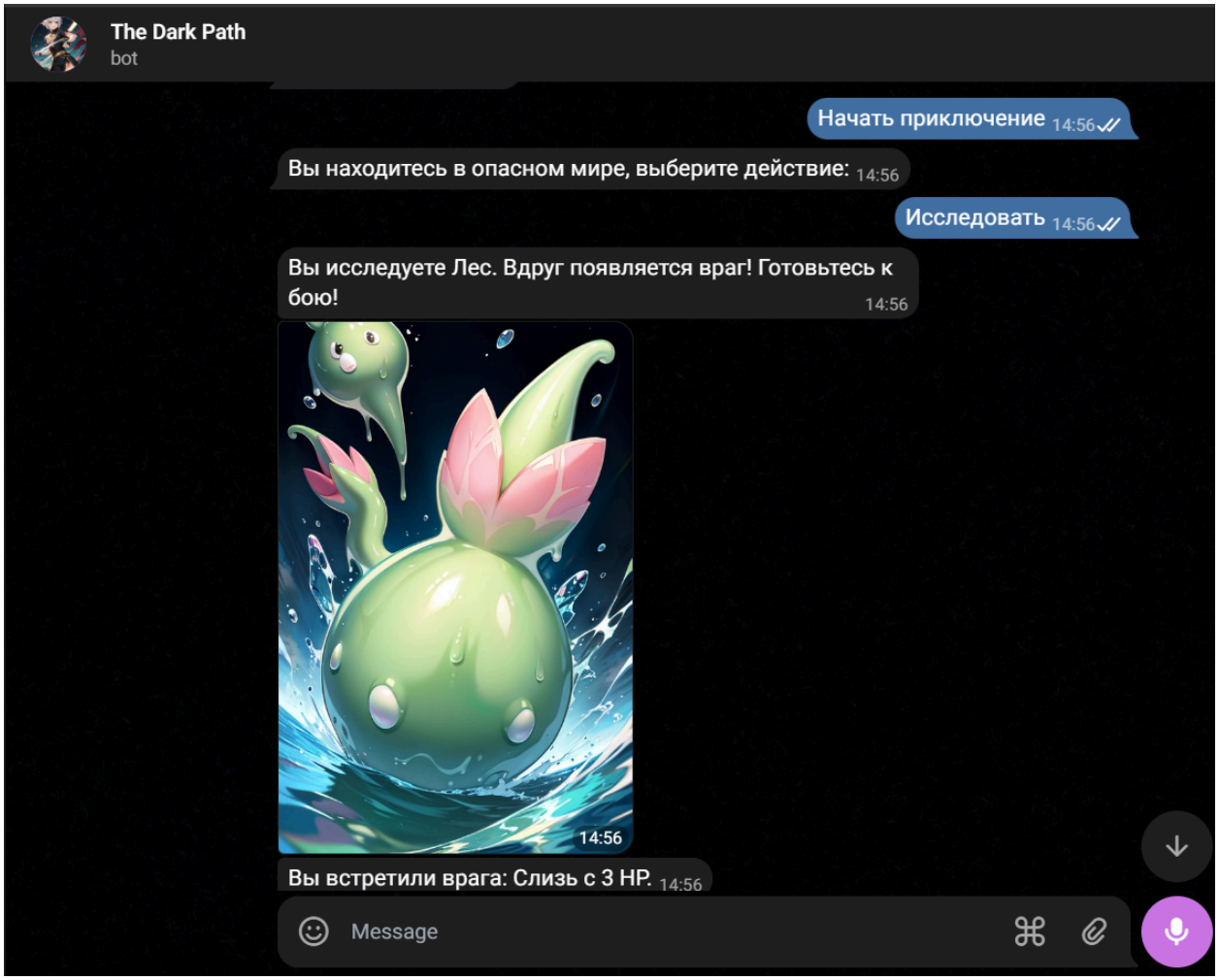
```
                    f"Ваше здоровье: {player['hp']}
HP\nУровень: {player['level']}\nОпыт: {player['xp']}")



@bot.message_handler(func=lambda message: message.text
== "Пройти обучение")

def training(message):

  xp_gain = random.randint(10, 30)



  player['xp'] += xp_gain



  bot.send_message(message.chat.id, f"Вы прошли
обучение! Получили {xp_gain} опыта.")



  if player['xp'] >= player['level'] * 100:

      player['level'] += 1

      player['xp'] = 0

      player['hp'] += 1

      player_hp += 1

      bot.send_message(message.chat.id,f"Поздравляем!
Вы достигли нового уровня! Теперь ваш уровень:
{player['level']}. Ваше здоровье увеличилось на 1!
Текущее здоровье: {player['hp']}.")



  send_welcome(message)
```

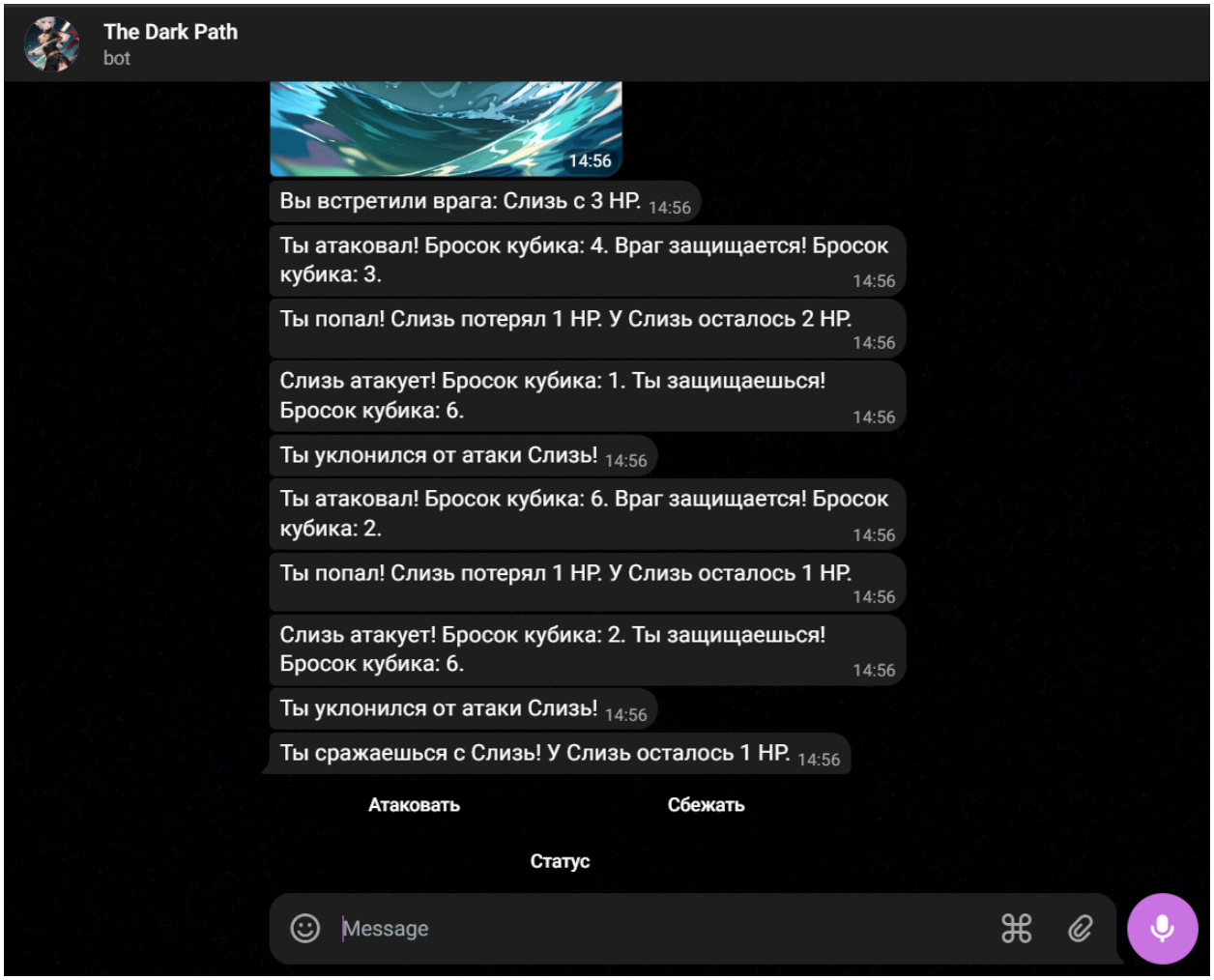
```
@bot.message_handler(func=lambda message: message.text
== "Начать приключение")

def adventure(message):

  send_adventure_options(message)




@bot.message_handler(commands=['start'])

def send_welcome(message):

  player['chat_id'] = message.chat.id

  markup =
types.ReplyKeyboardMarkup(resize_keyboard=True)

  item1 = types.KeyboardButton("Начать приключение")

  item2 = types.KeyboardButton("Проверить статус")

  item3 = types.KeyboardButton("Пройти обучение")

  markup.add(item1, item2, item3)


  bot.send_message(message.chat.id, "Привет, искатель
приключений! Готов начать свое путешествие?",

                   reply_markup=markup)


@bot.message_handler(func=lambda message: message.text
in ["Исследовать", "Искать приключения"])

def handle_action(message):

  action = message.text

  if action == "Исследовать":
```

```
        location = random.choice(["Лес", "Горы",
"Пещера", "Магическая поляна"])

        bot.send_message(message.chat.id, f"Вы
исследуете {location}. Вдруг появляется враг!
Готовьтесь к бою!")

        enemy = random.choice(enemies)

        bot.send_message(message.chat.id, f"Вы встретили
врага: {enemy['name']} с {enemy['hp']} HP.")

        battle(enemy, message)

    elif action == "Искать приключения":

        enemy = random.choice(enemies)

        bot.send_message(message.chat.id, f"Вы встретили
врага: {enemy['name']} с {enemy['hp']} HP.")

        battle(enemy, message)


if __name__ == "__main__":

    bot.polling(none_stop=True)
```

Экранные формы с примерами выполнения программы


The Dark Path
bot
/start 14:56
Привет, искатель приключений! Готов начать свое путешествие?
14:56
Проверить статус 14:56
14:56
Ваше здоровье: 3 HP
Уровень: 1
Опыт: 0
14:56
Message


The Dark Path
bot
Начать приключение 14:56
Вы находитесь в опасном мире, выберите действие: 14:56
Исследовать 14:56
Вы исследуете Лес. Вдруг появляется враг! Готовьтесь к бою! 14:56
14:56
Вы встретили врага: Слизь с 3 HP. 14:56
Message



Вывод

- Бот корректно реагирует на команды пользователя.
- Конечный автомат обеспечивает управление состояниями: бот меняет состояние в зависимости от действий пользователя (например, начальное состояние, исследование, тренировка).
- Реализованы ограничения, такие как лимит тренировок и проверка уровней.
- Программа соответствует поставленной задаче и демонстрирует использование конечного автомата для управления логикой бота.