



9 класс · Урок 25 из 34 · 45 минут

Программирование*: простой Telegram-бот через API

5.8. Некоторые приемы программирования для работы с искусственным интеллектом*

Результат: запускает проверяемую заготовку aiogram, прослеживает обработку сообщения и отличает офлайн-испытание кода от подключения живого бота.

Листы ученика

01. Проблемная ситуация

Программа написана правильно, но без учительского токена она не подключается к Telegram. Можно ли проверить обработчики и изучить API без регистрации ученика и без публикации секрета?

За 1 минуту прочитайте ситуацию; за 2 запишите предварительное решение и основание; за 2 сравните с соседом и назовите недостающую проверку. Случаи, числа и реплики условного ИИ созданы для обучения, не являются результатами испытания действующего сервиса.

Место для ответа



02. Объяснение и разобранный пример

5-14 минут: 4 минуты - понятия; 3 - совместный разбор; 2 - проверка ограничений примера.

API - правила обмена между программами. Бот получает обновление, диспетчер направляет его роутеру, фильтр выбирает подходящее событие, обработчик выполняет действие и отправляет ответ через API. Обычный бот по готовым правилам ещё не является нейросетью.

Токен разрешает доступ к конкретному боту. Его получает учитель для своего учебного бота через официальный BotFather, проверяет правила сервиса и не отправляет секрет в статью, чат, скриншот или репозиторий. Код читает BOT_TOKEN из переменной окружения. Ученики работают с подготовленной средой; отдельный аккаунт им не нужен.

asynсio запускает асинхронные функции. asynс def определяет такую функцию, await ожидает операцию без обязательного блокирования всех других задач. Это не команда «сделать всё одновременно» и не гарантия отсутствия ошибок. Точка входа - блок, запускающий main при выполнении файла, а не при его импорте.

Заготовка проверена 15 октября 2026 на Python 3.12 и aiogram 3.31.0: настоящая маршрутизация библиотеки испытана на шести искусственных обновлениях с заменой сетевого транспорта. Реальный токен и соединение Telegram при подготовке не использовались. Поэтому мы подтверждаем работу кода в офлайн-испытании, но не существование работающего публичного сервиса.

Разобранный пример. Сообщение /start проходит фильтр CommandStart и вызывает start. Обычный текст попадает в text_message, фотография - в other_message. Порядок обработчиков существенен: широкий обработчик без фильтра, поставленный первым, может перехватить команду. Наша программа не сохраняет и не повторяет личный текст пользователя.



03. Практика 1: Запустите минимальный код и проследите маршрут

Подготовленная папка содержит bot.py, offline.py и teacher_launch.py. Установка в отдельной среде: `python -m venv .venv`; затем её Python выполняет `python -m pip install aiogram==3.31.0`. Учитель делает это до урока и запускает `python offline.py` для проверки. Основной код ниже полностью совпадает с комплектом PDF. Python 3.10+ поддерживается библиотекой; здесь испытана версия 3.12.

```
import asyncio
import os
from aiogram import Bot, Dispatcher, F, Router
from aiogram.filters import Command, CommandStart
from aiogram.types import Message

def make_dispatcher():
    router = Router()

    @router.message(CommandStart())
    async def start(message: Message):
        await message.answer("Учебный бот. Команда /help.")

    @router.message(Command("help"))
    async def help_message(message: Message):
        await message.answer("Команды: /start и /help.")

    @router.message(F.text)
    async def text_message(message: Message):
        await message.answer("Текст получен. Личные данные не нужны.")

    @router.message()
    async def other_message(message: Message):
        await message.answer("Этот тип сообщения здесь не используется.")

    dispatcher = Dispatcher()
    dispatcher.include_router(router)
    return dispatcher

async def main():
    token = os.getenv("BOT_TOKEN")
    if not token:
        raise SystemExit("BOT_TOKEN задаёт учитель вне кода.")
    dispatcher = make_dispatcher()
    async with Bot(token=token) as bot:
        await dispatcher.start_polling(
            bot, allowed_updates=dispatcher.resolve_used_update_types()
        )

if __name__ == "__main__":
    asyncio.run(main())
```



Вход	Ожидаемое действие
/start	Учебный бот. Команда /help.
/help	Команды: /start и /help.
привет	Текст получен. Личные данные не нужны.
фотография	Этот тип сообщения здесь не используется.

- За 2 минуты найдите точку входа, диспетчер, роутер, фильтры и две строки обращения к API.
- За 4 минуты запустите `python offline.py` и сопоставьте четыре входа из таблицы с обработчиками. Запишите PASS только после реального вывода программы; ошибка тоже входит в протокол.
- За 3 минуты проследите цепочку /start и объясните, где данные сообщения превращаются в запрос SendMessage. При подготовленном учителем живом боте взрослый может дополнительно отправить /start и /help в свой учебный чат.
- За 2 минуты оформите протокол: среда, входы, ожидаемые/фактические ответы, офлайн или Telegram, что пока не проверено.

Форма результата: Таблица четырёх входов и протокол среды/статуса.

Критерии: По 1 баллу: элементы; реальный запуск/разбор ошибки; маршрутизация; честный протокол. Всего 4.

Место для ответа



04. Практика 2: Испытайте неизвестную команду и безопасный отказ

```
import asyncio
from datetime import datetime, timezone
from aiogram import Bot
from aiogram.client.session.base import BaseSession
from aiogram.methods import GetMe, SendMessage
from aiogram.types import Chat, Message, Update, User
from bot import make_dispatcher

class NoNetwork(BaseSession):
    def __init__(self):
        super().__init__()
        self.sent = []

    async def close(self):
        pass

    async def make_request(self, bot, method, timeout=None):
        if isinstance(method, GetMe):
            return User(id=123456, is_bot=True, first_name="Test",
                        username="ii_lesson_test_bot")
        if not isinstance(method, SendMessage):
            raise AssertionError("Неожиданный метод API")
        self.sent.append((method.chat_id, method.text))
        return Message(message_id=1, date=datetime.now(timezone.utc),
                        chat=Chat(id=method.chat_id, type="private"),
                        text=method.text)

    async def stream_content(self, url, **kwargs):
        raise AssertionError("Сеть запрещена в офлайн-тесте")
        if False:
            yield b""

async def check():
    # Фиктивный идентификатор только для конструктора теста, не токен API.
    session = NoNetwork()
    async with Bot(token="123456:NOT_A_REAL_TOKEN", session=session) as bot:
        dispatcher = make_dispatcher()
        cases = [
            ("/start", "Учебный бот. Команда /help."),
            ("/help", "Команды: /start и /help."),
            ("привет", "Текст получен. Личные данные не нужны."),
            (None, "Этот тип сообщения здесь не используется."),
            ("/start@ii_lesson_test_bot", "Учебный бот. Команда /help."),
            ("/unknown", "Текст получен. Личные данные не нужны."),
        ]
        for number, (text, expected) in enumerate(cases, 1):
            message = dict(message_id=number, date=0,
                            chat={"id":1,"type":"private"},
                            from_user={"id":2,"is_bot":False,"first_name":"Test"})
            if text is None:
                message["photo"] = [dict(file_id="test",file_unique_id="test",
                                          width=1,height=1)]
            else:
                message["text"] = text
            update = Update(update_id=number, message=Message(**message))
            before = len(session.sent)
            await dispatcher.feed_update(bot, update)
            assert len(session.sent) == before + 1, number
            assert session.sent[-1] == (1, expected), number
        print("PASS: 6 случаев, настоящая маршрутизация aiogram, без сети")
```



```
if __name__ == "__main__":  
    asyncio.run(check())
```

Новые случаи теста: /start@ii_lesson_test_bot и /unknown. В NoNetwork используется фиктивная строка только для конструктора; это не действительный токен API. Ни один вызов не выходит в сеть. Дополнительно рассмотрите запуск bot.py без BOT_TOKEN и идею поставить обработчик @router.message() первым.

- За 2 минуты предскажите ответы на два новых входа; объясните разницу между командой с адресатом и неизвестной командой.
- За 3 минуты запустите все шесть случаев и сравните фактический вывод с прогнозом; если меняете код, сначала сохраните отдельную копию.
- За 3 минуты объясните отказ без BOT_TOKEN и последствия широкого первого обработчика; предложите безопасную исправляющую проверку, не вставляя настоящий секрет в файл.

Форма результата: Шесть проверок, объяснение двух отказов и план исправления.

Критерии: По 1 баллу: два прогноза; запуск шести случаев; безопасный отказ; порядок и повторная проверка. Всего 4.

Место для ответа



05. Обсуждение: аргумент и возражение

За 2 минуты защитите тезис «Бот автоматизирует ответы». За 3 возразите против автоматической передачи личного сообщения в будущую нейросеть. За 2 сформулируйте предел этой версии.

Место для ответа

06. Индивидуальная проверка понимания

Код не читает BOT_TOKEN и содержит настоящий секрет прямо в тексте. Ученик хочет опубликовать файл после одного удачного /start. Назовите четыре нужных действия.

За 3 минуты запишите ответ; за минуту сверьте ключ и исправьте другим цветом; за последнюю минуту сдайте запись. По 1 баллу за каждый из четырёх содержательных элементов.

Место для ответа



Для учителя: ответы и критерии

9 класс · Урок 25. Программирование*: простой Telegram-бот через API

Подготовьте: экран или распечатки, тетради, ручки и калькулятор; компьютер с подготовленным Python и aiogram. Все таблицы, тексты и ключи находятся в статье и PDF. Ключи открывайте после самостоятельной попытки; ученические листы печатайте отдельно от раздела учителя.

Вариант без регистрации: работайте по готовым материалам и учебным журналам на экране или бумаге. Ученики не создают аккаунтов и не передают личных данных. Учебная имитация позволяет проверить рассуждение, но не заменяет настоящий эксперимент с сервисом.

До урока учитель устанавливает aiogram==3.31.0 в отдельную среду и проверяет offline.py. Для дополнительного живого испытания взрослый получает токен собственного учебного бота через официальный BotFather, проверяет возрастные условия и запускает teacher_launch.py. Ввод скрыт, токен остаётся только в окружении процесса; запуск останавливается Ctrl+C. Нужен один процесс polling, а не несколько копий. Урок и код полностью доступны без аккаунтов учеников; офлайн-тест не называют живым сервисом.

Время	Этап	Результат
0-5	Проблемная ситуация	Первое решение и основания
5-14	Теория и разобранный пример	Понятия, доказательства и ограничения
14-25	Практика 1	Проверяемый результат по критериям
25-33	Практика 2	Перенос в новые условия
33-40	Обсуждение	Аргумент, возражение, общее решение
40-45	Индивидуальная проверка	Самостоятельный ответ и сверка

Что собрать: результаты двух практик, основания вывода и индивидуальную проверку. Минуты включают чтение, обсуждение и запись; раздел соответствия программе предназначен учителю вне этих 45 минут. При 0-1 балле в проверке вернитесь к разобранному примеру; при 2-3 уточните пропущенное звено; при 4 предложите новый случай. Критерии нужны для обратной связи, не задают обязательную школьную отметку.

01-02. Разбор проблемной ситуации и примера

Разобранный пример. Сообщение /start проходит фильтр CommandStart и вызывает start. Обычный текст попадает в text_message, фотография - в other_message. Порядок обработчиков существенен: широкий обработчик без фильтра, поставленный первым, может перехватить команду. Наша программа не сохраняет и не повторяет личный текст пользователя.

03. Запустите минимальный код и проследите маршрут

Точка входа - if __name__ == "__main__"; диспетчер создан через Dispatcher, роутер через Router и включён include_router. CommandStart, Command("help") и F.text фильтруют события. message.answer формирует отправку ответа; в офлайн-тесте её исполняет NoNetwork. Четыре входа соответствуют таблице. PASS подтверждает проверенные пути библиотеки, не реальный токен, сеть или доступность Telegram.

Критерии: По 1 баллу: элементы; реальный запуск/разбор ошибки; маршрутизация; честный протокол. Всего 4.



04. Испытайте неизвестную команду и безопасный отказ

Команда с адресатом проверенного тестового бота вызывает start; /unknown попадает в обработчик текста и получает его фиксированный ответ. Без BOT_TOKEN main завершает запуск с понятным сообщением, не открывая соединение. Широкий первый обработчик может перехватить команды; исправление - сохранить специфические фильтры перед общим и повторить шесть тестов. Офлайн-проверка не проверяет настоящий секрет и сеть.

Запуск только учителем без записи секрета в файл: python teacher_launch.py. Файл teacher_launch.py полностью приведён ниже.

```
import asyncio
import getpass
import os
from bot import main

try:
    os.environ["BOT_TOKEN"] = getpass.getpass("BOT_TOKEN (только учитель): ")
    asyncio.run(main())
finally:
    os.environ.pop("BOT_TOKEN", None)
```

Критерии: По 1 баллу: два прогноза; запуск шести случаев; безопасный отказ; порядок и повторная проверка. Всего 4.

05. Обсуждение: ожидаемый результат

За 2 минуты защитите тезис «Бот автоматизирует ответы». За 3 возразите против автоматической передачи личного сообщения в будущую нейросеть. За 2 сформулируйте предел этой версии.

Бот ускоряет обмен по правилам, но не является универсальным ИИ. В этой версии не подключена языковая модель, не хранится переписка и не подтверждена реальная служба Telegram. Дальнейшее расширение требует отдельной проверки данных, разрешений, ошибок и фактических ответов.

Критерии: аргумент по материалу; содержательное возражение; условие итогового решения; роль человека. По 1 баллу, всего 4.

06. Индивидуальная проверка: ключ

Не публиковать секрет; учителю отозвать скомпрометированный токен через официальный инструмент и задать новый вне кода; читать переменную окружения; повторить разные тесты и отделить проверку маршрутизации от живого подключения. По 1 баллу за каждый содержательный пункт.



07. Связь с программой

Пункт программы	Действие	Проверяемый результат
Создавать бот на Python с использованием библиотеки <code>aiogram</code> и библиотеки асинхронного программирования <code>asyncio</code> .	Запускает подготовленный код	Шесть реальных офлайн-тестов настоящей библиотеки
Раскрывать смысл понятий «токен», «роутер», «хендлер», «точка входа», «фильтр».	Объясняет маршрут обновления	Связывает понятия со строками кода
Получать токен - уникальный код телеграм-бота.	Учитель подготавливает собственный учебный бот	Секрет вне публикации; фактический статус подключения
Писать минимальный код для запуска бота.	Разбирает <code>main</code> и точку входа	Запуск и безопасный отказ без переменной

Источники и права

Основа содержания и результатов - [исходная программа класса](#). Поурочное название и учебные случаи - авторская методическая развёртка, не дословные названия модулей программы. Числовые наборы, реплики условных помощников и ситуации специально подготовлены для урока. Учебные журналы не являются результатами испытания реального ИИ.

[aiogram 3.31.0: официальная документация](#); проверено 15 октября 2026 - источник для понятий и проверки; учебные задания разработаны отдельно.

[Telegram: официальный Bot API](#) - источник для понятий и проверки; учебные задания разработаны отдельно.

[Python 3.12: asyncio, официальная документация](#) - источник для понятий и проверки; учебные задания разработаны отдельно.

Логотип «Школы России» используется в материалах портала; авторские схемы предназначены для учебной работы. Внешние материалы сохраняют указанные условия использования.