🤖 Crypto Tech: Автоматизация и Алготрейдинг

ПРОДВИНУТЫЙ ТЕХНИЧЕСКИЙ МОДУЛЬ

Цель курса — обучить вас работе с API бирж, подключению вебхуков, настройке автоматических торговых скриптов, а также мониторингу объёмов и крупных кошельков (китов) без необходимости постоянного присутствия перед монитором. Данный курс представляет собой не теоретический обзор, а практическое техническое руководство для разработчиков, стремящихся автоматизировать криптотрейдинг на профессиональном уровне. Программа охватывает полный технологический стек: от построения инфраструктуры и обеспечения безопасности API до организации ончейн-мониторинга, управления рисками в программном коде и применения искусственного интеллекта для генерации и отладки торговых ботов.

МОДУЛЬ 1

🔌 Инфраструктура, API и Вебхуки

Прежде чем приступить к написанию первой строки кода торгового бота, необходимо чётко понимать, каким образом программный скрипт взаимодействует с биржей. В отличие от ручной торговли посредством веб-интерфейса или мобильного приложения, автоматизированные системы функционируют на принципиально ином уровне — уровне протоколов и API-запросов. Биржа предоставляет программный интерфейс (API), посредством которого ваш скрипт может выставлять ордера, проверять баланс, получать рыночные данные и управлять позициями — всё это без участия оператора и без использования графического интерфейса.

Существует два основных типа соединения с биржевым API: REST API и WebSocket. Каждый из них предназначен для решения определённого круга задач, и понимание различий между ними является критически важным условием для построения эффективной торговой системы. REST API функционирует по принципу «запрос — ответ»: ваш скрипт направляет HTTP-запрос к серверу биржи, сервер обрабатывает его и возвращает результат. Данный подход оптимально подходит для однократных операций — выставления ордера, проверки баланса, отмены заявки. WebSocket, напротив, устанавливает постоянное открытое соединение (стрим), посредством которого биржа непрерывно передаёт данные: тики, обновления стакана, изменения цен. Это решение незаменимо для получения рыночных данных в режиме реального времени с минимальной задержкой.

Задержка (latency) является ключевым параметром при выборе типа соединения. REST API характеризуется средней задержкой 100–300 мс, что приемлемо для большинства торговых задач, однако недостаточно для высокочастотных стратегий. WebSocket обеспечивает задержку менее 10–20 мс, что делает его единственным обоснованным выбором для стратегий, требовательных к скорости получения данных. На практике профессиональные торговые боты применяют оба типа соединения одновременно: WebSocket — для получения рыночных данных в режиме реального времени, REST API — для исполнения ордеров и управления позициями.

Ведущие биржи — Bybit, Binance, OKX — располагают подробной документацией по своим API. Каждый API имеет версию (v3, v5 и т.д.), набор эндпоинтов, лимиты частоты запросов (rate limits) и специфические форматы данных. До начала разработки настоятельно рекомендуется ознакомиться с официальной документацией выбранной биржи: в ней описаны все доступные методы, примеры запросов, коды ошибок и рекомендации по аутентификации. Большинство бирж также предоставляют тестовые среды (testnet), позволяющие отрабатывать интеграцию без риска для реальных средств.

REST API

Принцип работы: Запрос — Ответ (Request / Response)

Для чего используется:

- Выставить ордер
- Проверить баланс
- Отменить ордер
- Получить историю сделок

Задержка (Latency): Средняя — 100–300 мс

WebSocket (WS)

Принцип работы: Постоянное открытое соединение (Stream)

Для чего используется:

- Получать стакан цен в реальном времени
- Получать тики и обновления цен
- Отслеживать изменения позиций
- Мониторить ликвидацию ордеров

Задержка (Latency): Минимальная — < 10–20 мс

ⓘ 💡 **Ключевой принцип:** Торговый бот не использует графический интерфейс (UI). Он направляет запросы непосредственно к серверам биржи через API (Application Programming Interface). Это означает, что весь процесс торговли — от анализа рыночной ситуации до исполнения ордеров — осуществляется на уровне программного кода, без участия оператора и без применения визуального интерфейса.

МОДУЛЬ 1 — БЕЗОПАСНОСТЬ

🔐 Безопасность API-ключей: OpSec для разработчика

Безопасность API-ключей — это не рекомендация, а обязательное требование. При создании API-ключа на бирже (Bybit, Binance, OKX) вы получаете два значения: **API Key** (публичный) и **API Secret** (приватный). Публичный ключ используется для идентификации вашего запроса, а приватный секрет — для подписи каждого запроса цифровой подписью (HMAC-SHA256). Если злоумышленник получит доступ к вашему API Secret, он сможет торговать от вашего имени, а при наличии соответствующих разрешений — и вывести все средства с биржи.

История криптотрейдинга знает множество случаев, когда разработчики вшивали API-ключи прямо в код, коммитили их в публичные репозитории GitHub, и их средства были украдены в течение нескольких часов. Даже если вы работаете локально и не планируете выкладывать код в интернет, привычка хранить ключи в коде — это технический долг, который рано или поздно приведёт к инциденту. Используйте файл окружения `.env` с самого начала проекта — это стандарт индустрии для управления секретами в Python-проектах.

Библиотека `python-dotenv` позволяет загружать переменные окружения из файла `.env` прямо в ваш Python-скрипт. Файл `.env` должен быть добавлен в `.gitignore`, чтобы никогда не попасть в систему контроля версий. В коде вы обращаетесь к ключам через `os.getenv("API_KEY")` — никогда напрямую через строковые литералы. Этот подход также упрощает развёртывание: на продакшн-сервере (VPS) вы просто создаёте свой `.env` с реальными ключами, а в репозитории хранится только шаблон `.env.example` без секретов.

IP-привязка — ещё один критически важный уровень защиты. Большинство бирж позволяют привязать API-ключ к конкретному IP-адресу. Это означает, что даже если ключ будет скомпрометирован, злоумышленник не сможет использовать его с другого IP. При развёртывании бота на VPS-сервере (например, Hetzner, DigitalOcean, AWS) установите статический IP и привяжите к нему все API-ключи. Если вы разрабатываете локально, используйте IP вашего домашнего подключения или настройте VPN с фиксированным выходным IP.

▶️ Правила защиты API-ключей

🔒 Ограничение прав

Включайте разрешения **ТОЛЬКО** на «Торговлю» (Trade). **НИКОГДА** не включайте опцию «Вывод средств» (Enable Withdrawals). Данное правило не имеет исключений — даже если вы являетесь единственным, кто имеет доступ к серверу. В случае компрометации ключа отсутствие права на вывод защитит ваши средства от мгновенного опустошения.

🌐 IP-Привязка

Привязывайте API-ключ к статическому IP-адресу вашего сервера (VPS). Это создаёт дополнительный барьер: даже при утечке ключа злоумышленник не сможет использовать его с другого адреса. Настройте статический IP на своём VPS и укажите его в настройках безопасности биржи при создании ключа.

📁 Хранение в коде

Никогда не вшивайте ключи открытым текстом в скрипты. Используйте файл окружения `.env` и библиотеку `python-dotenv`. Добавьте `.env` в `.gitignore`, чтобы случайно не закоммитить секреты в репозиторий. В репозиторий добавляйте только шаблон `.env.example` без реальных значений.

⚠️ **⚠️ Критическое предупреждение:** API-ключ с правами на вывод средств — это фактически пароль от вашего кошелька. Если подобный ключ окажется в чужих руках, биржа не сможет оказать помощь — транзакции необратимы. Всегда создавайте ключи с минимально необходимыми правами.

МОДУЛЬ 2

📊 Базовые Алгоритмы: Сеточные Боты (Grid Bots)

Сеточный алгоритм (Grid Bot) — один из наиболее популярных и понятных торговых алгоритмов для работы в боковом движении (флэте / флэтовой консолидации). Идея заключается в следующем: бот расставляет сетку лимитных ордеров на покупку ниже текущей цены и на продажу выше текущей цены, создавая «коридор», внутри которого колеблется цена. Каждое пересечение уровня сетки генерирует сделку и фиксирует небольшую прибыль. Стратегия особенно эффективна на рынках с высокой волатильностью, но без выраженного тренда — когда цена движется в определённом диапазоне.

Настройка сеточного бота требует определения трёх ключевых параметров: верхней границы сетки, нижней границы сетки и количества уровней (гридов). Чем больше уровней — тем чаще будут исполняться сделки, однако тем меньше прибыль с каждого уровня. Оптимальное количество гридов зависит от волатильности актива и ширины ценового коридора. Для BTC/USDT в диапазоне $59 000–$65 000 типичное значение составляет 6–12 уровней. Необходимо также принимать во внимание комиссии биржи: если прибыль с одного грида меньше комиссии за сделку, стратегия окажется убыточной даже при идеальном боковом движении.

Сеточные боты сопряжены с определёнными рисками. Основным из них является выход цены за пределы установленного коридора. Если цена пробивает нижнюю границу, бот остаётся с открытой длинной позицией и незаполненными ордерами на покупку. Если цена пробивает верхнюю границу — бот продаёт весь актив и остаётся в USDT, упуская дальнейший рост. Поэтому перед запуском необходимо проанализировать историческую волатильность актива и установить достаточно широкий коридор, способный выдержать типичные колебания рынка.

Существуют два основных типа сеток: арифметическая (равные ценовые шаги между уровнями) и геометрическая (равные процентные шаги). Арифметическая сетка проще в настройке, однако геометрическая более устойчива при широких диапазонах цен, поскольку каждый уровень представляет одинаковый процентный шаг. Большинство современных биржевых платформ и библиотек (например, Hummingbot, Freqtrade) поддерживают оба типа сеток.

Пример сетки для BTC/USDT

[Верхняя граница сетки]
$65,000 ——————————— Sell Order 3
$64,000 ——————————— Sell Order 2
$63,000 ——————————— Sell Order 1
[Текущая цена] $62,000 ————————————————————————————
$61,000 ——————————— Buy Order 1
$60,000 ——————————— Buy Order 2
[Нижняя граница сетки]
$59,000 ——————————— Buy Order 3

Механика

Бот расставляет сетку лимитных ордеров на покупку ниже текущей цены и на продажу выше текущей цены. Каждый уровень сетки представляет собой отдельный лимитный ордер с заранее заданной ценой и объёмом. При достижении ценой соответствующего уровня ордер исполняется автоматически.

Заработок

Извлечение прибыли из микроколебаний цены внутри заданного коридора. Каждая пара «покупка-продажа» на соседних уровнях генерирует небольшую прибыль, которая накапливается за множество циклов.

Идеальные условия

Боковое движение (флэт / флэтовая консолидация) — рынок без выраженного тренда, при котором цена колеблется в диапазоне. На трендовых рынках сеточные боты демонстрируют менее эффективные результаты.

МОДУЛЬ 2

📈 DCA-Боты: Dollar-Cost Averaging в Алготрейдинге

DCA-бот (Dollar-Cost Averaging) — стратегия автоматического усреднения позиции при движении цены против первоначального входа. В отличие от сеточного бота, который работает в боковике, DCA-стратегия предназначена для трендовых ситуаций, когда трейдер уверен в направлении движения, но стремится оптимизировать точку входа. Бот открывает стартовую позицию, а затем при неблагоприятном движении цены добавляет к ней дополнительные ордера, снижая среднюю цену входа.

Ключевые параметры DCA-бота: размер стартового ордера, шаг усреднения (в процентах от цены), количество страховочных ордеров (Safety Orders) и размер каждого последующего ордера. Существует два подхода к масштабированию: линейный (каждый Safety Order равен стартовому) и мартингейл (каждый следующий ордер больше предыдущего). Мартингейл позволяет быстрее снизить среднюю цену входа, однако требует значительно большего капитала и несёт повышенный риск.

Автоматический расчёт уровня Take-Profit — критически важная функция DCA-бота. При каждом усреднении бот должен пересчитывать среднюю цену входа и устанавливать новый уровень TP таким образом, чтобы при отскоке цены закрыть всю совокупную позицию в плюс. Формула расчёта: **TP = Средняя цена входа × (1 + Целевая прибыль%)**. Чем больше Safety Order'ов исполнено, тем ниже средняя цена и тем меньший отскок необходим для выхода в прибыль.

Риск-менеджмент в DCA-ботах имеет особое значение, поскольку стратегия по своей природе предполагает увеличение позиции при убытке. Рекомендуется устанавливать максимальное количество Safety Order'ов и жёсткий стоп-лосс на случай, если цена продолжит движение против позиции после исчерпания всех усреднений. Без данных ограничений один сильный тренд против позиции может привести к значительным потерям.

Алгоритм работы DCA-бота

01

Стартовый ордер

Бот открывает стартовый ордер (например, Long по $60,000). Это базовая позиция, с которой начинается торговля. Размер стартового ордера обычно составляет 30–50% от общего выделенного капитала на стратегию.

02

Первый Safety Order

Если цена падает на 1.5%, бот исполняет первый страховочный ордер (Safety Order), снижая среднюю цену входа. Размер Safety Order может быть равен стартовому (линейный DCA) или превышать его (мартингейл).

03

Повторные усреднения

При каждом усреднении уровень Take-Profit автоматически смещается ниже, позволяя закрыть всю позицию в плюс при минимальном отскоке цены вверх. Бот продолжает добавлять Safety Orders до достижения максимального количества.

04

Закрытие позиции

При достижении обновлённого уровня Take-Profit бот закрывает всю совокупную позицию и фиксирует прибыль. После этого цикл может начаться заново при появлении нового сигнала на вход.

💡 DCA-стратегия эффективна на рынках с временными коррекциями в рамках общего восходящего тренда. Однако при сильном нисходящем тренде без отскоков DCA-бот может исчерпать весь выделенный капитал. Настоятельно рекомендуется устанавливать максимальное количество усреднений и жёсткий стоп-лосс.

МОДУЛЬ 3

🐋 On-Chain Мониторинг: Отслеживание Всплесков Объёмов

Резкий аномальный рост объёма на фьючерсах нередко предшествует сильному импульсному движению цены. Это один из наиболее надёжных технических сигналов, который поддаётся автоматизации с помощью простого Python-скрипта. Логика детекции основана на сравнении текущего минутного объёма со скользящим средним за предыдущий период (как правило, 20 минут). Если текущий объём превышает среднее значение в N раз (пороговое значение, или threshold), скрипт генерирует алерт.

Пороговое значение (threshold) является ключевым параметром алгоритма. Типичные значения составляют 2.0–3.0 для умеренных сигналов и 5.0+ для экстремальных аномалий. Чем выше порог — тем реже поступают сигналы, однако тем выше их значимость. Слишком низкий порог приводит к значительному количеству ложных срабатываний. Оптимальное значение подбирается экспериментально для каждого актива и таймфрейма. Для BTC/USDT на 1-минутном графике порог 3.0 обычно обеспечивает 3–8 сигналов в сутки, большинство из которых соответствуют реальным импульсным движениям.

Для получения данных об объёмах допускается использование WebSocket-стрима биржи (Bybit, Binance) или публичного REST API. WebSocket является предпочтительным вариантом, поскольку обеспечивает минимальную задержку. Библиотека `ccxt` (CryptoCurrency eXchange Trading Library) предоставляет унифицированный интерфейс для работы с десятками бирж и поддерживает как REST, так и WebSocket подключения. Для разработчиков, начинающих работу в данной области, `ccxt` существенно упрощает интеграцию, поскольку не требует изучения специфичных форматов данных каждой биржи.

Полученные сигналы об аномальном объёме могут быть использованы различными способами: для отправки уведомления в Telegram, автоматического открытия позиции или в качестве фильтра для других стратегий. Необходимо принимать во внимание, что всплеск объёма сам по себе не указывает направление движения — он лишь сигнализирует о повышенной активности крупных участников рынка. Для определения направления требуется дополнительный анализ цены и стакана ордеров.

Пример логики детекции всплеска объёма на Python

```
# Пример логики детекции всплеска объема на Python

import time

def check_volume_spike(current_volume, average_volume, threshold=3.0):
 """
 Сравнивает текущий минутный объем со средним за прошлый период.
 Если объем превышает норму в N раз, возвращает True.
 """
 if current_volume >= (average_volume * threshold):
 return True
 return False

# Использование

curr_vol = 1500000 # Текущий минутный объем в USDT
avg_vol = 300000 # Средний минутный объем за последние 20 минут

if check_volume_spike(curr_vol, avg_vol):
 print("
```

МОДУЛЬ 3

🐋 On-Chain Мониторинг Китов (Whale Alert)

Крупные игроки и фонды оставляют «цифровые следы» в блокчейне задолго до того, как их действия отражаются на цене. Перед массовой продажей токенов киты переводят монеты с холодных кошельков на биржи (чтобы иметь возможность осуществить продажу). Перед накоплением — напротив, выводят активы с бирж на собственные кошельки. Отслеживание этих перемещений в режиме реального времени предоставляет трейдеру существенное преимущество: возможность наблюдать намерения крупных участников рынка до того, как рынок на них отреагирует.

On-chain мониторинг представляет собой анализ транзакций непосредственно в блокчейне, а не на бирже. В отличие от биржевых данных, блокчейн-данные являются публичными, неизменяемыми и не могут быть скрыты от наблюдения. Любой адрес кошелька, любая транзакция доступна для анализа через обозреватели блоков (Etherscan, BscScan, Solscan) или специализированные API. Скрипт может отслеживать конкретные адреса кошельков или осуществлять мониторинг всех крупных транзакций по определённому токену.

Для автоматизации on-chain мониторинга существуют специализированные сервисы и API. **Nansen** — лидер в области аналитики «умных денег»: данный сервис маркирует кошельки по категориям (фонды, маркет-мейкеры, инсайдеры) и предоставляет API для программного доступа. **Arkham Intelligence** предлагает бесплатный API для отслеживания крупных транзакций и деанонимизации кошельков. **DeBank** специализируется на портфельной аналитике и отслеживании активности DeFi-кошельков. Для разработчиков, начинающих работу в данной области, Arkham обеспечивает наиболее доступный вход благодаря бесплатному тарифному плану.

Помимо мониторинга биржевых потоков, продвинутые скрипты способны отслеживать «умные кошельки» (Smart Money Wallets) — адреса с исторически высокой прибыльностью. Концепция заключается в автоматическом копировании действий трейдеров, демонстрирующих стабильно положительные результаты. Сервисы Nansen и Arkham уже идентифицировали тысячи подобных кошельков и присвоили им соответствующие метки. Ваш скрипт может быть настроен на получение уведомлений о транзакциях этих кошельков и использование их в качестве торговых сигналов.

Что мониторить через скрипты

Inflow на биржи

Массовый приток монеты на CEX → **Сигнал о возможном дампе / продаже.** Крупные участники рынка готовят продажу, переводя активы на биржу. Рекомендуется вести мониторинг адресов крупных держателей и биржевых депозитных адресов.

Outflow с бирж

Массовый вывод монеты с CEX → **Сигнал о накоплении и удержании.** Крупные участники рынка выводят активы с биржи, снижая давление продаж. Данный сигнал является бычьим для среднесрочного и долгосрочного горизонта планирования.

Smart Money Wallets

Отслеживание кошельков с высокой исторической прибыльностью через сервисы **Nansen**, **Arkham** или **DeBank**. Рекомендуется автоматически копировать действия трейдеров со стабильно положительной историей PnL.

🔍 **Рекомендация:** Сервис **Whale Alert** (@whale_alert в Twitter/X) публикует крупные транзакции в режиме реального времени. Для разработчиков доступен соответствующий API — его можно подключить непосредственно к собственному боту и настроить получение уведомлений о транзакциях, превышающих заданный пороговый уровень.

МОДУЛЬ 4

🛡️ Управление Рисками в Коде: Автоматический Kill-Switch

Даже идеально протестированный алгоритм может столкнуться с аномальной волатильностью, сбоем API биржи или непредвиденным поведением рынка. Именно поэтому в код любого торгового бота в обязательном порядке включается жёсткий лимит убытков — механизм, известный как Kill-Switch. Это не опция и не рекомендация — это обязательный элемент любой автоматизированной торговой системы, который должен быть реализован до первого реального запуска.

Kill-Switch функционирует по следующему принципу: скрипт непрерывно отслеживает совокупный PnL (прибыль/убыток) всех открытых позиций и накопленный дневной результат. Как только убыток достигает заданного порога (например, 3% от депозита), механизм срабатывает автоматически — без ожидания подтверждения от разработчика. Это критически важно, поскольку в моменты экстремальной волатильности специалист может не успеть своевременно среагировать, а бот продолжит торговать, усугубляя потери.

Реализация Kill-Switch в коде требует трёх компонентов: (1) функция расчёта текущего PnL по всем позициям, (2) переменная-счётчик дневного убытка, которая сбрасывается в полночь по UTC, и (3) функция экстренного закрытия, которая через REST API биржи закрывает все позиции по рынку и отменяет активные ордера. После срабатывания скрипт должен установить флаг блокировки, предотвращающий открытие новых сделок в течение заданного периода (как правило, 24 часа).

Помимо дневного лимита убытков, профессиональные торговые системы включают дополнительные уровни защиты: лимит на максимальный размер позиции, лимит на количество одновременных открытых сделок, максимальную просадку (max drawdown) и таймауты на повторное подключение к API. Все указанные ограничения должны быть настроены до запуска и протестированы на тестовой сети биржи.

🛑 Предохранитель (Hard Stop)

> Если суммарный дневной убыток бота превышает **3% от депозита**, скрипт автоматически:
>
> 1. Закрывает все открытые позиции по рынку (market close)
> 2. Отменяет все активные ордера
> 3. Блокирует запуск новых сделок на 24 часа

📊 Мониторинг PnL

Непрерывный расчёт совокупного PnL всех открытых позиций в реальном времени. Скрипт опрашивает биржу каждые несколько секунд и суммирует нереализованный PnL по всем позициям.

⚡ Автоматическое исполнение

При достижении порога — незамедлительное закрытие через REST API без ожидания подтверждения. Каждая миллисекунда задержки способна обернуться дополнительными процентами убытка.

🔒 Блокировка 24ч

После срабатывания — принудительная пауза для анализа ситуации и предотвращения повторных потерь. Скрипт фиксирует причину остановки в лог-файле для последующего разбора.

⊗ 🛑 **Критически важно:** Kill-Switch — это последняя линия обороны. Он не заменяет корректное управление размером позиции и грамотную настройку стратегии. Если Kill-Switch срабатывает с высокой частотой — это сигнал к пересмотру параметров бота, а не к увеличению лимита убытков.

МОДУЛЬ 5

🤖 ИИ-Копилот в Разработке Ботов

Современные языковые модели (ChatGPT, Claude, Gemini) стали незаменимыми инструментами для разработчиков торговых ботов. Они способны генерировать рабочий код, выявлять уязвимости, предлагать архитектурные решения и объяснять сложные концепции доступным языком. Однако ключ к эффективному использованию ИИ — это грамотно сформулированные промты (запросы). Качественный промт задаёт контекст, роль, конкретную задачу и формат ожидаемого результата. Некорректно составленный промт даёт расплывчатый, практически бесполезный ответ.

Первый промпт демонстрирует, как использовать ИИ для написания кода с нуля. Вы задаёте модели роль Senior Python Developer, описываете конкретную задачу (мониторинг объёма Bybit и Telegram-уведомления), указываете библиотеки и пороговые значения. Результат — готовый рабочий скрипт, требующий минимальной доработки. Данный подход позволяет сэкономить значительное время и сосредоточиться на логике стратегии, а не на синтаксисе API.

Второй промпт показывает, как использовать ИИ для аудита кода и поиска уязвимостей. Вы вставляете существующий код и запрашиваете у модели выявление проблем, добавление обработки ошибок и реализацию Kill-Switch. Это особенно полезно для начинающих разработчиков, которые могут быть не осведомлены о специфичных ошибках биржевых API (таймауты, rate limits, коды ошибок). ИИ также способен предложить улучшения архитектуры: добавление логирования, повторных попыток подключения и graceful shutdown.

При работе с ИИ для генерации торгового кода необходимо всегда проверять полученный результат. Модели могут «галлюцинировать» — генерировать несуществующие методы API или использовать устаревшие параметры. Сгенерированный код следует обязательно сверять с официальной документацией биржи. Кроме того, категорически не рекомендуется передавать ИИ реальные API-ключи или конфиденциальные данные — используйте плейсхолдеры (`YOUR_API_KEY`) в примерах кода.

Промпт №1: Написание Python-бота для Telegram-уведомлений

"Вы — Senior Python Developer. Напишите простой скрипт на Python с использованием библиотеки requests, который подключается к публичному REST API Bybit, получает текущий 1-минутный объем торгов по BTCUSDT и отправляет уведомление в Telegram-бота через Telegram Bot API, если объем за минуту превышает $2,000,000."

Промпт №2: Поиск багов и обработка ошибок API

"Проанализируйте данный Python-код торгового бота [Вставить код]. Выявите уязвимости, добавьте обработку ошибок таймаутов API (HTTP 502/504) через try-except и реализуйте функцию Kill-Switch, которая закрывает все позиции при потере 3% депозита."

Когда использовать Промпт №1

- Начало нового проекта с нуля
- Интеграция нового API или сервиса
- Создание шаблонов уведомлений
- Генерация boilerplate-кода

Когда использовать Промпт №2

- Аудит существующего кода
- Поиск уязвимостей безопасности
- Добавление обработки ошибок
- Рефакторинг и оптимизация

🛠️ ИНСТРУМЕНТЫ И ФИНАЛЬНОЕ ЗАДАНИЕ

🏁 Практическое Итоговое Задание по Курсу 5

Теория без практики не имеет ценности. Финальное задание курса объединяет все изученные модули в один рабочий проект: Telegram-бот для мониторинга объёмов с автоматическими алертами. Это реальный инструмент, который вы можете использовать в своей торговой деятельности сразу после завершения курса. Задание разбито на четыре последовательных шага — каждый следующий опирается на результат предыдущего.

Для выполнения задания вам понадобится: аккаунт на Bybit (или другой бирже с публичным API), установленный Python 3.8+, библиотека `requests` (устанавливается через `pip install requests`), и аккаунт Telegram. Все инструменты являются бесплатными. Для запуска скрипта можно использовать локальный компьютер, однако для круглосуточной работы рекомендуется бесплатный облачный сервер — **PythonAnywhere** или **Replit** позволяют запускать Python-скрипты без дополнительной оплаты.

После выполнения задания у вас будет полностью рабочий мониторинговый бот, который можно расширять: добавить детекцию нескольких торговых пар, интегрировать WebSocket для получения данных в реальном времени, подключить on-chain мониторинг крупных участников рынка или добавить автоматическое исполнение сделок. Это ваш фундамент для построения полноценной автоматизированной торговой системы.

📄 Готовый скрипт: Отправка алерта в Telegram (Python)

```
import requests

def send_telegram_alert(bot_token, chat_id, message):
 url = f"https://api.telegram.org/bot{bot_token}/sendMessage"
 payload = {
 "chat_id": chat_id,
 "text": message,
 "parse_mode": "HTML"
 }
 try:
 response = requests.post(url, json=payload)
 return response.json()
 except Exception as e:
 print(f"Ошибка отправки алерта: {e}")

# Пример использования:
# send_telegram_alert("YOUR_BOT_TOKEN", "YOUR_CHAT_ID",
# "
```