Geekbrains

**Автоматизация торговли на крипто валютной бирже и создание**

**телеграм - робота помощника для контроля фьючерсного аккаунта**

Цифровые профессии:

Python - разработка

Кротов С.В.

Пермь, 2024

**Содержание**

# 

[**Глава 1: Введение 3**](#_gjkk3mq26suz)

[**1.1 Актуальность выбранной темы 3**](#_cvrz9ngw2qol)

[**1.2 История развития 3**](#_93bwgrns1izz)

[**1.3 Цели и задачи работы 5**](#_tvsbkz67qhtr)

[**1.4 Основные понятия торгового бота 5**](#_9xftb62x5g29)

[**1.5 Анализ существующих решений на рынке 6**](#_8i6n75zcv4n0)

[**1.6 Методы и инструменты разработки торгового бота на языке Python 7**](#_8xkgm52fbm83)

[**Глава 2: Основы автоматизации торговли на крипто валютных биржах 8**](#_ifsf9n8i5zk5)

[**2.1 Обзор крипто валютных бирж и их API 8**](#_bbdtvadgeme9)

[**2.2 Методы технического анализа и торговые стратегии 11**](#_jmn82jatrazg)

[**2.3 Безопасность и управление рисками при торговле криптовалютами 14**](#_5gzcsan8cbgb)

[**2.4 Математические модели и методы машинного обучения для анализа рынка 15**](#_6hbcxcz0w4ea)

[**Глава 3: Проектирование и разработка торгового бота 16**](#_1lux53r3kmip)

[**3.1 Разработка модулей для взаимодействия с API криптовалютной биржи 17**](#_b35ouxoze03w)

[**3.2 Реализация торговых стратегий и методов управления портфелем 23**](#_jxrxk8w4c21m)

[**Глава 4: Разработка телеграмм-робота помощника. 27**](#_5c5uj4iv3u35)

[**4.1 Теоретические основы 27**](#_v7rql2zi6qu6)

[**4.2 Анализ существующих решений 29**](#_aeg2hsd3ef8j)

[**4.3 Разработка телеграмм-робота помощника 30**](#_4eu7pxl8xf56)

[**4.4 Выбор библиотек и изучение API биржи для фьючерсной торговли 31**](#_rhioriht4jcg)

[**Глава 5: Создание телеграмм-робота для контроля фьючерсного аккаунта 33**](#_moc3gb1ffkt6)

[**5.1 Интеграция с торговым ботом и отображение данных о фьючерсном счете 39**](#_s82hvg49v01c)

[**5.2 Тестирование и оптимизация робота для контроля фьючерсного аккаунта 44**](#_7bcdvqn7gx6z)

[**Глава 6: Заключение 48**](#_hgaq9jlgtu1u)

[**6.1 Выводы по исследованию 48**](#_pdnff0z9e7f7)

[**6.2 Перспективы дальнейших исследований в данной области 49**](#_gg2r81tvve51)

# 

# Глава 1: Введение

# 1.1 Актуальность выбранной темы

Создание торгового робота на онлайн крипто валютной бирже очень актуально в современном мире финансовых технологий. С развитием криптовалют и возрастанием интереса к ним, количество торговых операций на крипто биржах значительно увеличилось. В связи с этим, многие трейдеры и инвесторы ищут новые способы автоматизации торговли, чтобы увеличить свои шансы на успешные сделки.

Торговые роботы предлагают целый ряд преимуществ, таких как постоянный мониторинг рынка, быстрые реакции на изменения цен, отсутствие эмоционального влияния на принятие решений и возможность торговать круглосуточно. Учитывая высокую волатильность крипто валютных рынков, использование торгового робота может дать трейдерам значительное преимущество и защитить их от больших потерь.

Более того, создание своего собственного торгового робота на онлайн криптобирже позволяет управлять своим капиталом и стратегией торговли, исходя из индивидуальных потребностей и предпочтений. Это дает возможность персонализировать автоматическую торговлю в соответствии с уникальными целями и стилем торговли каждого трейдера.

Таким образом, в условиях динамичного и конкурентного рынка криптовалют использование торгового робота на онлайн бирже становится все более востребованным и актуальным. Он помогает трейдерам улучшить свои результаты торговли, минимизировать риск и получить дополнительные возможности для заработка.

# 1.2 История развития

История развития и сущность криптовалюты открывает увлекательный мир цифровых денег, начавший формироваться задолго до появления Bitcoin. Эволюция криптовалют проходила через несколько ключевых этапов, отражающих растущую потребность в децентрализованных и безопасных средствах обмена.

Криптовалюта — это цифровая или виртуальная валюта, использующая криптографию для обеспечения безопасности, создания новых единиц и верификации передачи активов. Основная особенность криптовалюты — она не контролируется ни одним центральным органом, что предотвращает любые манипуляции или вмешательство правительства.

Необходимость в создании цифровых денег обсуждалась ещё в 1980-х годах. Работы Дэвида Чаума, в том числе его статья "Blind signatures for untraceable payments" (1982), положили начало разработке анонимных цифровых валют. Основываясь на этих идеях, в 90-е годы появляются первые проекты, такие как DigiCash, однако они так и не смогли достичь широкого распространения.

Рождение Bitcoin

В 2008 году появляется статья под псевдонимом Сатоши Накамото "Bitcoin: A Peer-to-Peer Electronic Cash System", которая описывает концепцию децентрализованной цифровой валюты без центрального органа. Собственно, в январе 2009 года был запущен Bitcoin, первая криптовалюта, положившая начало новой эре в истории финансов.

Развитие экосистемы и альтернативные криптовалюты. С появлением Bitcoin начинает складываться новая индустрия с собственной инфраструктурой: биржами, кошельками, сервисами. Появление блокчейн-технологии, сердце Bitcoin, стимулировало появление альтернативных криптовалют (альткойнов), таких как Litecoin (2011), Ripple (2012), Ethereum (2015). Каждая новая криптовалюта стремилась решить определенные проблемы Bitcoin или предложить новые функции, например, более быстрые транзакции или "умные контракты" в случае Ethereum.

Бум ICO и регулирование, 2017 год запомнился "бумом" первичных предложений монет (ICO), когда множество проектов собрали миллиарды долларов на разработку своих платформ и приложений. Однако растущее внимание регуляторов и множество мошеннических проектов привели к ужесточению контроля и потребности в легитимизации криптовалют в глазах правительств и финансовых учреждений.

Несмотря на значительные колебания стоимости и постоянное давление со стороны государственных регуляторов, криптовалюты и технология блокчейн продолжают развиваться, предлагая инновационные решения в финансовой сфере, логистике, праве и многих других областях. Устойчивый интерес к криптовалютам подчеркивает их потенциал сформировать основу будущей глобальной финансовой системы.

# 1.3 Цели и задачи работы

Целью данного дипломного проекта является разработка и реализация системы автоматизации торговли на крипто валютной бирже, а также создание телеграмм-робота помощника для контроля фьючерсного аккаунта.

Основной задачей работы является создание программного обеспечения, которое позволит автоматизировать процесс торговли на крипто валютной бирже. Это включает в себя разработку алгоритмов для анализа рыночной ситуации, принятия решений о покупке и продаже активов, а также мониторинг за выполнением торговых стратегий и управление рисками.

Другой важной задачей проекта является создание телеграмм-робота, который будет выполнять функцию помощника трейдера и контролировать фьючерсный аккаунт. Этот робот будет предоставлять информацию о текущем состоянии счета, открытых позициях, риск-параметрах, а также отправлять уведомления о важных событиях на рынке.

Таким образом, целью работы является повышение эффективности торговли на крипто валютной бирже за счет автоматизации процессов и создание инструмента для удобного контроля за фьючерсным аккаунтом с помощью телеграмм-робота.

# 1.4 Основные понятия торгового бота

Основные понятия торгового бота для криптовалютной биржи включают:

1. API (Интерфейс программирования приложений): Это набор протоколов и инструментов, предоставляемых биржей для взаимодействия с торговым ботом. С помощью API бот может получать данные о биржевой котировке, размещать ордера на покупку и продажу, а также осуществлять другие действия.

2. Технический анализ: Это метод анализа ценовых графиков и объемов торгов для принятия решений о покупке или продаже криптовалюты. Торговые боты обычно используют различные индикаторы и стратегии технического анализа для автоматического прогнозирования рыночной динамики.

3. Автоматизированные стратегии торговли: Торговые боты могут осуществлять торговлю на основе заранее заданных правил и параметров, таких как уровень стоп-лосс, уровень тейк-профит, размер ордера и прочее. Это позволяет боту автоматически реагировать на изменения рыночной ситуации без участия трейдера.

4. Риск-менеджмент: Это ключевой аспект для торговли криптовалютами. Торговый бот должен уметь управлять риском, определять размер позиции и устанавливать уровни стоп-лосс, чтобы минимизировать потенциальные убытки.

5. Бэктестинг (Backtesting): Это процесс тестирования торговой стратегии на исторических данных. Торговый бот должен иметь возможность проводить бэктестинг, чтобы оценить эффективность стратегии на прошлых данных перед ее использованием в реальном времени.

Это основные понятия, которые должен учитывать торговый бот при работе на крипто валютной бирже.

# 1.5 Анализ существующих решений на рынке

На рынке существует множество решений для создания торговых ботов для крипто валютных бирж. Некоторые из них представляют собой готовые программные решения, другие — платформы для разработки собственных ботов.

Среди популярных готовых решений можно выделить следующие:

1. 3commas: Платформа, предоставляющая возможность создания торговых ботов с использованием различных стратегий, включая технический анализ, стоп-лосс и тейк-профит ордера.

2. HaasOnline: Помимо возможности создания торговых ботов, HaasOnline предлагает богатое разнообразие инструментов для технического анализа и автоматизированной торговли на крипто валютных рынках.

3. Cryptohopper: Это платформа, позволяющая создавать торговых ботов с использованием стратегий анализа рынка на основе индикаторов технического анализа.

4. Zenbot: Открытый и доступный для изменения проект на Node.js, Zenbot предоставляет возможность создания собственных торговых стратегий и ботов.

Также есть платформы, которые предоставляют инструменты для разработки собственных торговых ботов, такие как CryptoTrader и Gunbot. Они предлагают гибкость настройки и позволяют трейдерам реализовывать свои уникальные стратегии торговли.

Каждое из этих решений имеет свои преимущества и недостатки, и выбор оптимального инструмента зависит от требований конкретного трейдера. Кроме того, важно учитывать вопросы безопасности и надежности при использовании любых программных решений на рынке криптовалют.

# 1.6 Методы и инструменты разработки торгового бота на языке Python

Существует множество методов и инструментов разработки торгового робота для криптовалютной биржи на языке Python. Вот несколько популярных методов и инструментов:

1. API биржи: Большинство крипто валютных бирж предоставляют API для взаимодействия с их платформой. Вы можете использовать библиотеки Python, такие как ccxt (CryptoCurrency eXchange Trading Library), для удобного взаимодействия с API различных бирж.

2. Технический анализ: Существует множество библиотек Python для технического анализа, таких как TA-Lib, PyAlgoTrade, backtrader и др. Они предоставляют широкий выбор индикаторов и удобные инструменты для анализа ценовых данных.

3. Расчет риска и управление портфелем: Для реализации методов управления рисками и портфелем можно использовать математические библиотеки Python, такие как NumPy и pandas, для выполнения расчетов и анализа данных.

4. Интеграция с торговыми стратегиями: Python предлагает широкие возможности для разработки и реализации торговых стратегий благодаря своей гибкости и широкому спектру библиотек для анализа данных.

5. Библиотеки машинного обучения: Для создания прогностических моделей и использования методов машинного обучения можно использовать библиотеки Python, такие как scikit-learn, TensorFlow, Keras и другие.

6. Развертывание и автоматизация: Для развертывания и автоматизации торгового бота можно использовать различные инструменты и фреймворки, такие как Docker, Celery, Airflow и другие.

Объединяя эти инструменты, мы сможем разработать мощный и гибкий торговый бот для криптовалютной биржи на языке Python.

# Глава 2: Основы автоматизации торговли на крипто валютных биржах

# 2.1 Обзор крипто валютных бирж и их API

Обзор крипто валютных бирж и их API важен для понимания доступных инструментов для автоматизации торговли криптовалютой.

API крипто валютных бирж предоставляет программный интерфейс для взаимодействия с биржей и может включать в себя методы для получения данных о рынке, управления ордерами, выполнения торговых операций, управления средствами на счете и многие другие функции.

При выборе криптовалютной биржи для автоматизированной торговли важно учитывать не только наличие API, но и его функциональность, стабильность, поддержку и безопасность.

Перед использованием API крипто валютных бирж важно ознакомиться с их документацией, ограничениями, требованиями к безопасности и возможностями, чтобы правильно интегрировать и использовать их в своих системах автоматизированной торговли.

Ниже представлен обзор популярных криптовалют ных бирж и их API:

1. Binance:

- Binance - одна из крупнейших и наиболее популярных криптовалют ных бирж в мире.

- У Binance есть API, предоставляющее доступ к торговым данным, выполнению торговых операций, получению истории заказов и т.д.

2. Coinbase:

- Другая крупная криптовалютная биржа, специализирующаяся на купле-продаже криптовалюты.

- Coinbase также предоставляет API с различными методами для работы с аккаунтом, ордерами и рыночными данными.

3. Kraken:

- Kraken известна своими продвинутыми функциями и отличной репутацией в сообществе криптовалют.

- У них также есть API, которое позволяет получить доступ к различным торговым функциям и данным о рынке.

4. Bitfinex:

- Bitfinex предлагает широкий спектр функций для профессиональных трейдеров.

- Их API позволяет получить доступ к информации о торгах, управлению аккаунтом, созданию ордеров и другим функциям.

5. Huobi:

- Одна из ведущих крипто валютных бирж в Азии с широким функционалом для торговли.

- У них также есть API для интеграции с их платформой.

6. Bybit:

- криптовалютная биржа, специализирующаяся на торговле криптовалютными деривативами, такими как фьючерсы и опционы. Биржа предлагает широкий спектр торговых пар, включая BTC/USD, ETH/USD, XRP/USD и другие.

Bybit также предоставляет API (Application Programming Interface) для автоматизации торговли и взаимодействия с платформой. Их API предоставляет различные методы для получения маркетинговых данных, управления ордерами, выполнения торговых операций и управления аккаунтом.

Некоторые особенности и возможности API Bybit включают:

1. Получение рыночных данных: API позволяет получать информацию о текущих ценах, объемах торгов и других маркетинговых данных.

2. Управление ордерами: API позволяет создавать, отменять и изменять торговые ордера на бирже.

3. Исполнение торговых операций: С помощью API можно выполнять покупку и продажу крипто валютных деривативов.

4. Управление аккаунтом: API предоставляет возможности для управления средствами на счете, проверки баланса и истории торговых операций.

5. Защита и безопасность: Bybit обеспечивает безопасность API с помощью различных методов аутентификации и авторизации, чтобы защитить аккаунты пользователей.

Использование API криптовалютной биржи Bybit позволяет трейдерам автоматизировать торговлю, создавать торговые стратегии, использовать алгоритмическое трейдинговые программы.

Однако перед использованием API Bybit важно ознакомиться с их документацией, правилами использования и безопасностью API, чтобы правильно интегрировать его в свои автоматизированные торговые стратегии.

# 2.2 Методы технического анализа и торговые стратегии

При торговле на онлайн крипто валютных биржах часто используются различные методы технического анализа и торговые стратегии для выявления трендов, определения точек входа и выхода из рынка, а также управления рисками. Ниже приведены основные методы технического анализа и торговые стратегии, которые широко применяются при торговле криптовалютами на биржах:

Методы технического анализа:

1. Графики цен:

- Использование различных типов графиков, таких как свечные, линейные и баровые, для анализа движения цены и выявления трендов.

2. Индикаторы технического анализа:

- Использование различных технических индикаторов, таких как скользящие средние, MACD, RSI и другие, для определения точек входа и выхода из рынка.

3. Паттерны ценового движения:

- Выявление классических паттернов ценового движения, таких как "голова и плечи", "флаг", "треугольник" и другие, для прогнозирования будущего движения цены.

Торговые стратегии:

1. Breakout (Прорыв):

- Торговая стратегия, основанная на разрыве цены через уровень поддержки или сопротивления.

2. Trend following (Поддержка тренда):

- Стратегия, основанная на следовании за установившимся трендом и входе в рынок в направлении тренда.

3. Mean reversion (Возврат к среднему):

- Стратегия, основанная на предположении, что цена имеет тенденцию вернуться к своему среднему значению, и входе в рынок на основе этого предположения.

4. Arbitrage (Арбитраж):

- Стратегия, основанная на одновременной покупке и продаже одного и того же актива на разных биржах для извлечения прибыли из разницы в ценах.

Важно отметить, что выбор методов технического анализа и торговых стратегий зависит от индивидуальных предпочтений трейдера, его стиля торговли и уровня опыта. Также важно учитывать риск-менеджмент и использовать стоп-лоссы для защиты от потенциальных убытков.

В ходе данной работы мною был выбран индикатор RSI

**Relative** **strength** **index** - индекс **относительной** **силы** (RSI) является одним из наиболее популярных индикаторов технического анализа, который широко используется при торговле на онлайн крипто валютных биржах.

RSI измеряет скорость и изменение ценовых движений актива, сравнивая величину последних положительных и отрицательных изменений цен. Индикатор RSI выражается в виде числового значения от 0 до 100. Часто считается, что актив перекуплен, если RSI превышает 70, и перепродан, если RSI опускается ниже 30.

При торговле на криптовалютных биржах RSI может использоваться для следующих целей:

1. Определение перекупленности и перепроданности рынка: Торговцы могут использовать RSI для определения момента, когда актив может быть перекуплен или перепродан, что может сигнализировать о возможном развороте тренда.

2. Подтверждение тренда: Различные уровни RSI могут использоваться для подтверждения текущего тренда. Например, если актив находится в установившемся восходящем тренде, RSI может подтвердить силу этого тренда, когда значение индикатора поднимается выше уровня 50.

3. Идентификация дивергенций: RSI может использоваться для выявления дивергенций между ценой актива и значением индикатора, что может сигнализировать о возможном развороте тренда.

Таким образом, RSI может быть полезным инструментом для определения точек входа и выхода из рынка, а также для оценки силы текущего тренда. Однако, как и любой другой индикатор, RSI следует использовать в сочетании с другими методами технического анализа и торговыми стратегиями, а также учитывать риск-менеджмент для принятия обоснованных торговых решений.

# 2.3 Безопасность и управление рисками при торговле криптовалютами

Торговля криптовалютами на онлайн биржах может быть крайне рискованным делом, поскольку криптовалютные рынки обладают высокой волатильностью и подвержены значительным колебаниями. Поэтому безопасность и управление рисками играют важную роль в процессе торговли. Ниже приведены некоторые ключевые меры безопасности и принятые методы управления рисками при торговле криптовалютами на онлайн биржах:

1. Использование надежных бирж:

- При выборе биржи для торговли криптовалютами необходимо убедиться в ее надежности, репутации и безопасности. Исследуйте и изучите рейтинги и отзывы других трейдеров о бирже, а также убедитесь, что она регулируется соответствующими органами и обеспечивает высокий уровень защиты средств и данных клиентов.

2. Использование двухфакторной аутентификации (2FA):

- Включите двухфакторную аутентификацию на своем аккаунте на бирже, чтобы обеспечить дополнительный уровень безопасности при доступе к вашему аккаунту.

3. Хранение средств в холодных кошельках:

- Для безопасного хранения крупных сумм криптовалют, предпочтительно использовать холодный кошелек (cold wallet), который находится вне интернета и более устойчив к хакерским атакам.

4. Управление рисками:

- Разработка и строгое соблюдение индивидуальной стратегии управления рисками, включающей установку стоп-лоссов, ограничение размера позиций, диверсификацию портфеля и другие меры, которые помогут минимизировать потенциальные убытки.

5. Ознакомление с законодательством и налогами:

- Прежде чем приступить к торговле, важно изучить фискальное и правовое положение в нашей стране относительно криптовалютных операций, чтобы избежать неприятных юридических последствий.

6. Обучение и постоянное обновление знаний:

- Инвестировать в свое образование и становиться информированным трейдером, следя за последними тенденциями на крипто валютных рынках и изучая новые методы управления рисками.

И, наконец, помнить, что торговля криптовалютами, особенно на онлайн биржах, несет свои риски и требует дисциплинированного подхода и осторожного планирования.

# 2.4 Математические модели и методы машинного обучения для анализа рынка

В анализе рынка криптовалют на онлайн биржах часто используются различные математические модели и методы машинного обучения. Некоторые из них включают в себя:

1. Временные ряды: анализ временных рядов может помочь предсказать будущие изменения цен криптовалют на основе исторических данных.

2. Методы глубокого обучения: нейронные сети могут использоваться для анализа больших объемов данных и выявления сложных закономерностей на рынке криптовалют.

3. Методы анализа сентимента: анализ сообщений в социальных сетях и новостей может помочь оценить общее настроение рынка и его влияние на цены криптовалют.

4. Алгоритмы машинного обучения: такие как случайные леса, градиентный бустинг и другие, могут применяться для прогнозирования цен криптовалют и определения оптимальных стратегий торговли.

5. Методы портфельной оптимизации: оптимизация портфеля может помочь инвесторам распределить свои активы между различными криптовалютами таким образом, чтобы минимизировать риски и максимизировать доходность.

Эти методы и модели могут быть успешно применены для анализа рынка криптовалют и принятия решений о торговле на онлайн биржах. Важно помнить, что торговля криптовалютами всегда связана с высоким уровнем риска, поэтому необходимо внимательно и осмотрительно подходить к использованию этих методов и прогнозов.

# Глава 3: Проектирование и разработка торгового бота

При проектировании архитектуры бота для торговли криптовалютами на онлайн биржах на языке Python важно учитывать не только функциональные требования, но и аспекты производительности, безопасности, масштабируемости и надежности.

1. Модуль взаимодействия с API биржи: этот модуль отвечает за взаимодействие с API биржи для получения данных о рынке, отправки ордеров на покупку/продажу и выполнения других операций. Можно использовать готовые библиотеки, такие как ccxt, для упрощения работы с API различных бирж.

в моей работе этот модуль называется config.py в нем будут храниться ключи доступа к аккаунту на бирже

2. Модуль стратегии торговли: здесь определяются правила входа и выхода из сделок, управление рисками, оптимизация параметров стратегии и другие аспекты вашей торговой стратегии. Разработка эффективной стратегии является одним из ключевых аспектов успешной торговли.

данный модуль он же и основной назыается main.py

3. Модуль выполнения ордеров: этот модуль отвечает за отправку и выполнение ордеров на бирже в соответствии с выбранной стратегией торговли. Необходимо обеспечить надежность и безопасность выполнения ордеров, а также учитывать важные аспекты, такие как минимальный объем сделки, комиссии и прочее.

модуль имеет название orders.py

4. Модуль управления данными: этот модуль отвечает за сбор и обработку данных о рынке, исторических данных, анализ рынка и другие операции, необходимые для принятия решений торгового бота.

модуль имеет название settings.py

# 3.1 Разработка модулей для взаимодействия с API криптовалютной биржи

# 

Давайте приступим. Создаем виртуальное окружение. Итак, сначала мы делаем нужные импорты. Будем использовать библиотеку Pybit

Установить его можно вот такой командой pip install pybit. Импортируем HTTP для того чтобы создать нашу сессию. Всю подробную информацию по взаимодействию с биржей берем из оффициальной документации на сайте: <https://bybit-exchange.github.io/docs/v5/intro>

в модуль config.py мы указываем api key и secret key. В модуль settings.py мы указываем symbol - название торговой пары которой мы будем торговать и kline-time эта переменная отвечает за таймфрейм(свечи) цифрой указывается количество минут. Код будет асинхронным потому что:

1. Эффективность: асинхронное программирование позволяет выполнять множество задач параллельно без блокировки потоков, что повышает эффективность использования ресурсов и ускоряет выполнение операций.

2. Отзывчивость: использование асинхронного кода позволяет легко обрабатывать асинхронные события, такие как получение данных от API биржи или выполнение большого количества запросов в одно время, что делает ваш бот более отзывчивым на изменения на рынке.

3. Управление несколькими соединениями: торговые боты часто требуют одновременного общения с несколькими биржами или сервисами. Асинхронное программирование облегчает управление множеством соединений и запросов одновременно.

4. Повышение производительности: благодаря эффективному использованию ресурсов и параллельной обработке задач, асинхронный код может улучшить производительность вашего торгового бота, особенно при работе с большим объемом данных.

5. Простота разработки: асинхронное программирование с использованием async/await синтаксиса в Python 3.5 и выше делает разработку более удобной и понятной для программистов, особенно при работе с сетевыми запросами и обработкой большого объема данных.

Использование асинхронного программирования не является необходимым для создания торгового робота, но при выполнении операций, которые требуют множественного параллельного взаимодействия с биржами или сервисами, асинхронный подход может значительно улучшить производительность и эффективность моего робота.

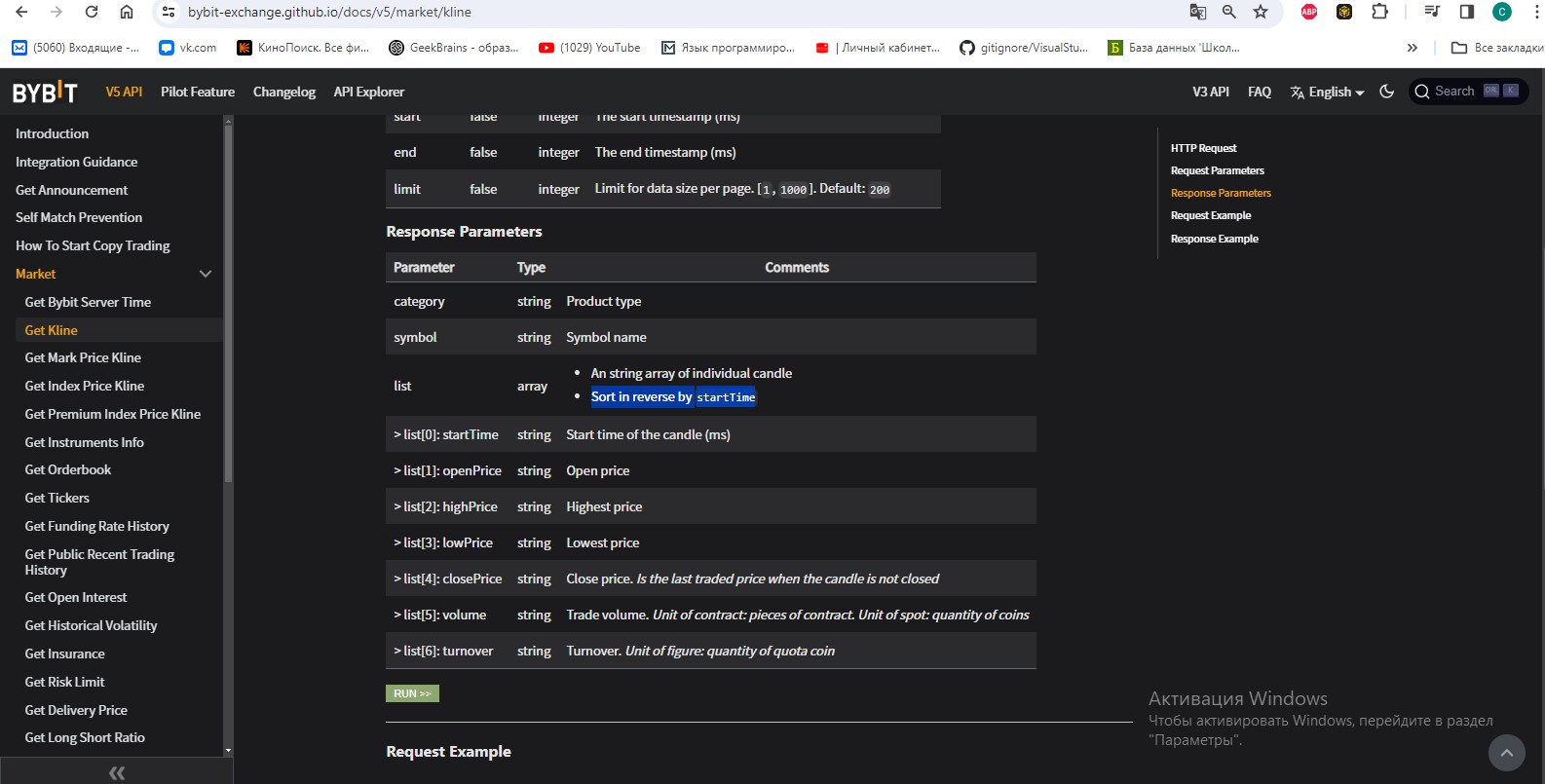
Напишем конструкцию if name == 'main' а также делаем импорт asyncio

Создаю бесконечный цикл while True, в котором я буду запрашивать наши свечи, который будем анализировать с помощью индикатора RSI. Поставлю также обязательно sleep в 60 сек

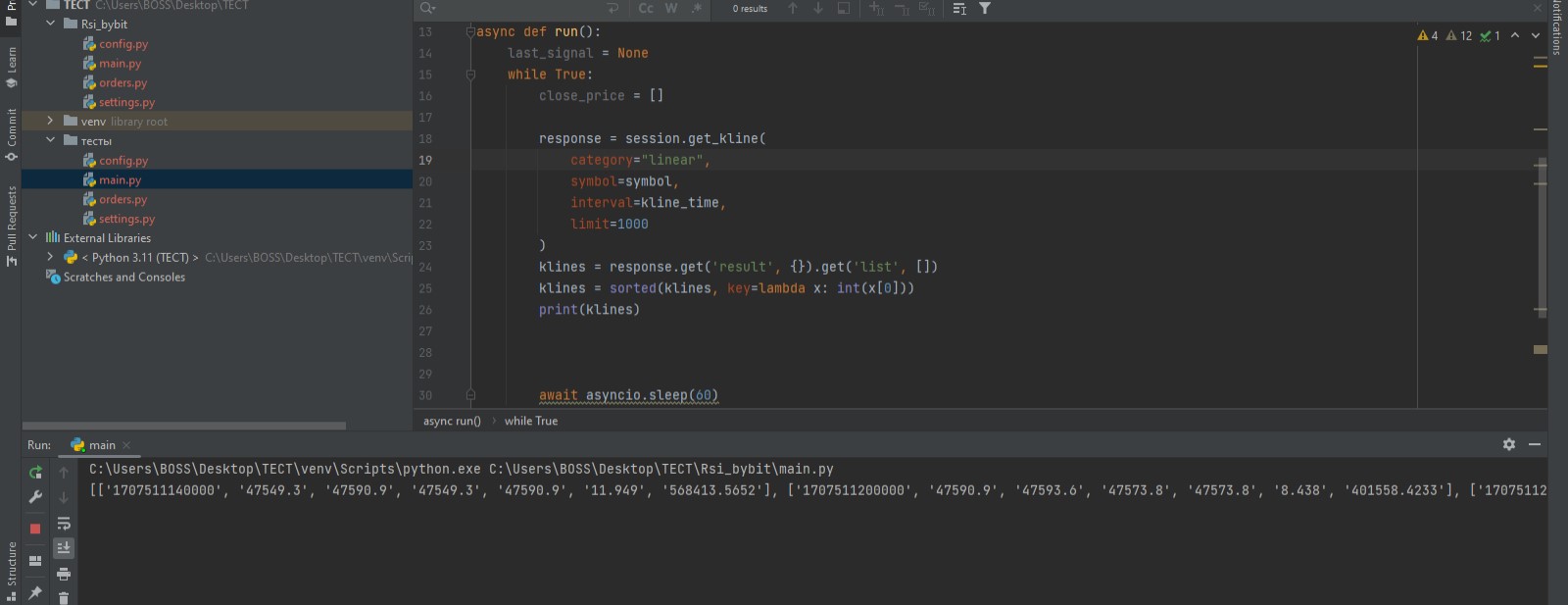
Делаю запрос на свечи, указываю категорию, с которой мы запрашиваем свечи, category="linear" это указывает, что это фьючерсы, limit=1000 это количество свечей, которые мы запрашиваем, в нашем случае это 1000. Если этот параметр не указывать, то стандартно будет запрашиваться 200 свечей, мы можем указать 1000, это максимум для данной биржи.

klines = response.get('result', {}).get('list', [])

klines = sorted(klines, key=lambda x: int(x[0]))

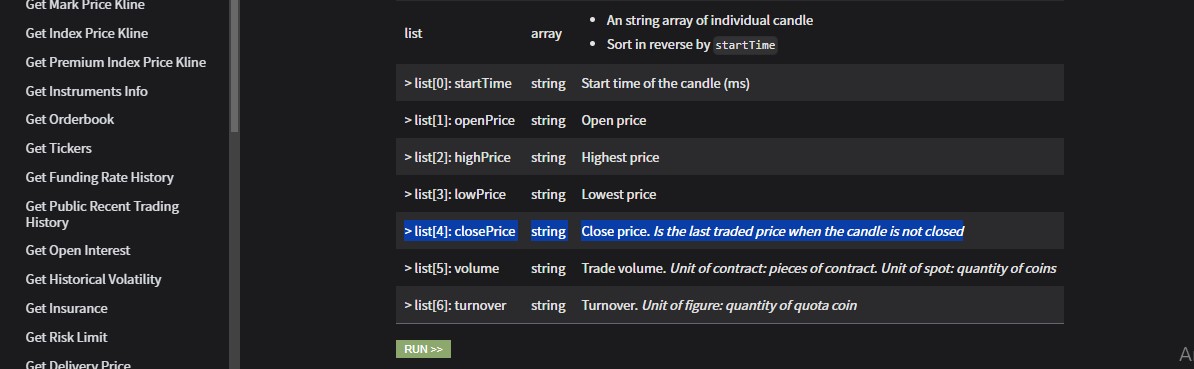
Итак, эту сортировку я делаю потому, что если мы посмотреть в документацию Bybit - там написано Sort in reverse by start time. 

Это говорит о том, что свечи на бирже Bybit по данному запросу возвращаются в обратном порядке. То есть самая актуальная свеча будет в начале списка, а не в его конце, как, например, на Binance. Это важный момент. Поэтому я пишу данную сортировку, чтобы получать в нужном нам порядке свечи. Итак, давайте проверим это.

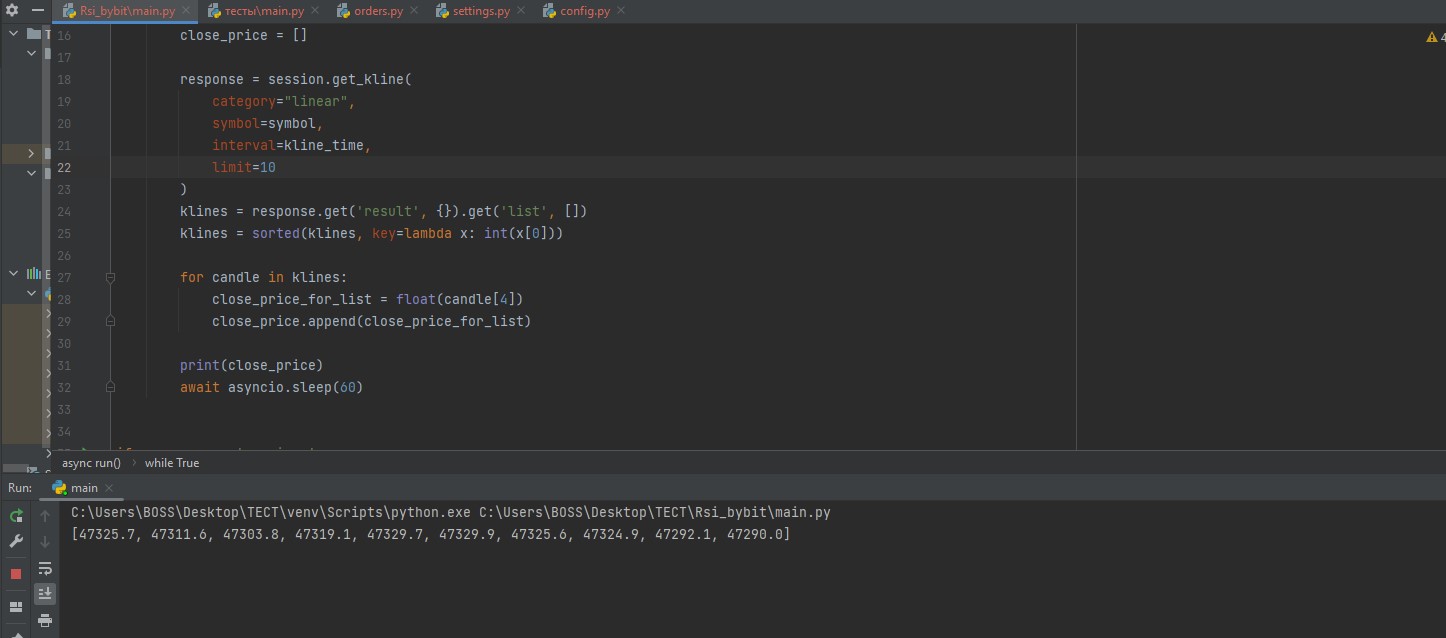
Просто выведем klines, все ли у нас на данном этапе хорошо срабатывает.

Да, все работает

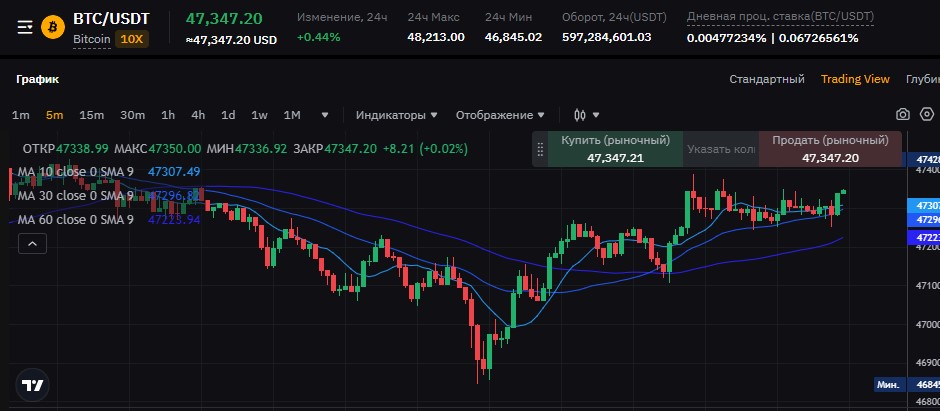
Мы получили все свечи, здесь пока у нас не только close prices, которые нам нужны, то есть цены закрытия, а здесь у нас вся информация по свечам. Вот мы можем заметить в документации ее. То есть наш start time, open price, high price и так далее. Итак, сейчас нам нужно получить только close prices. Для этого мы создали пустой список, в котором будем добавлять цены закрытия.



Просто создаем цикл, который перебирает все наши свечи и обращается к четвертому элементу свечи. Давайте посмотрим. Так как у нас счет идет от нуля, четвертый имеем Close Price. Добавляем в список через append наши цены закрытия и давайте выведем, посмотрим. У нас есть список со всеми ценами закрытия. Давайте может даже сравним их.



и

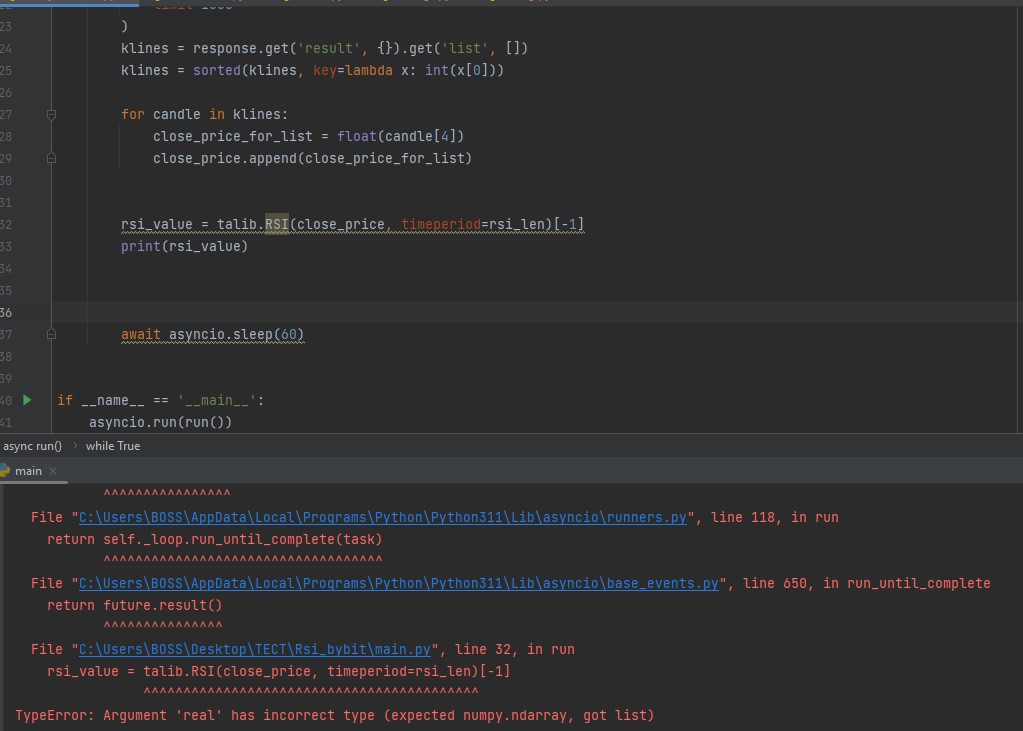


Давайте выведем список, посмотрим на актуальную цену. 47,347. Ну да, 47,325, тут цена прыгает. Но мы поняли, что мы получаем актуальную цену в конце списка, то есть теперь у нас все это работает правильно.

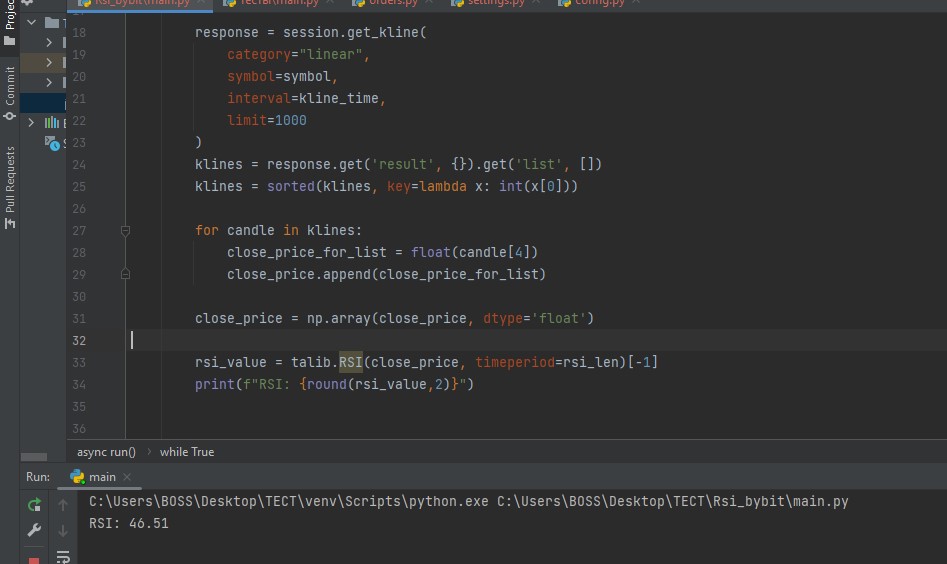
Теперь нам нужно наши Close Prices отправлять в индикатор RSI для анализа, буду использовать библиотеку talib. Делаем импорт библиотеки talib. Установка этой библиотеки отличается от простого pip install, необходимо знать свою версию python и разрядность системы, найти и скачать whl файл, а затем положить его в в папку lib либо в папку с проектом и потом установить его из этой папки.

rsi\_value = talib.RSI(close\_price, timeperiod=rsi\_len)[-1]

Здесь я передаю сам индикатор RSI в talib, указываем наш список цен закрытия и указываем период для нашего индикатора RSI. Давайте его в настройках укажем в модуле settings.py, сделаю его импорт и пусть он будет 14. И получаем последнее наше актуальное значение, так как RSI Value будет список из всех наших Close Price, то есть по всем ценам закрытия мы получим наш RSI. И нам нужен самый актуальный, то есть последний. И давайте его выведем.



Мы не можем передавать в RSI индикатор обычный список. Нам нужно его конвертировать из обычного списка в NumPy список. Делаю import numpy Данную строку я добавил, она конвертирует обычный список в numpy. Также мы всем значениям даем тип float. И просто передаем этот же список в наш индикатор. для более удобного вывода мы сделаем здесь RSI. И также давайте его округлим немного, чтобы он не писал много знаков после запятой. Получаем значение 46.5



Видим такое же значение на самом индикаторе на бирже Bybit 46.3

# 3.2 Реализация торговых стратегий и методов управления портфелем

При настройке торгового робота для торговли на криптобирже с использованием индикатора RSI, следует учитывать следующие моменты:

1. Длительность периода: Настройте период RSI в соответствии со своими торговыми стратегиями. Обычно используются значения от 14 до 21 дней.

2. Уровни сигналов: Установите уровни перекупленности и перепроданности на графике RSI (обычно 70 и 30 соответственно) для определения момента входа или выхода из сделки.

В модуле settings зададим значения, например, rsi\_low = 30 и rsi\_high = 70. rsi\_len - длительность периода уже указана

Итак, давайте напишем небольшую стратегию. Если rsi\_value меньше, чем rsi\_low, то мы делаем покупку (Buy). В случае, если rsi\_value больше, чем rsi\_high, мы делаем продажу (Sell).

И сейчас добавим ордера. Ордер на криптобирже – это инструкция, которую трейдер отправляет на биржу для совершения определенной операции с криптовалютой. Ордер указывает, какую сделку нужно выполнить (покупку или продажу), количество криптовалюты, по которому желает произвести операцию, а также цену, по которой он готов совершить сделку.

В ней мы будем передавать нашу сессию, символ(выбранную торгуемую пару), side(покупка или продажа) и цена закрытия.

async def market\_order(session,symbol,side,close\_price):

qty\_p, limit\_p = getPrecision(symbol, session)

qty = round(deposit / close\_price[-1], qty\_p)

market\_order = session.place\_order(

category="linear",

symbol=symbol,

side=side,

orderType="Market",

qty=qty,

timeInForce="PostOnly",

orderFilter="Order",

)

print(market\_order)

Чтобы рассчитать quantity, нам понадобится дополнительная функция, которая будет считать количество знаков для округления.

def getPrecision(symbol,session):

response = session.get\_instruments\_info(

category="linear", # or "spot", "inverse", "option" depending on your requirement

symbol=symbol # specify the trading pair

)

priceFilter = response['result']['list'][0]['priceFilter']

lotSizeFilter = response['result']['list'][0]['lotSizeFilter']

tickSize = priceFilter['tickSize']

qtyStep = lotSizeFilter['qtyStep']

if tickSize.startswith("0."):

tickSize = tickSize.replace("0.", "")

else:

tickSize = str(float(tickSize))

if qtyStep.startswith("0."):

qtyStep = qtyStep.replace("0.", "")

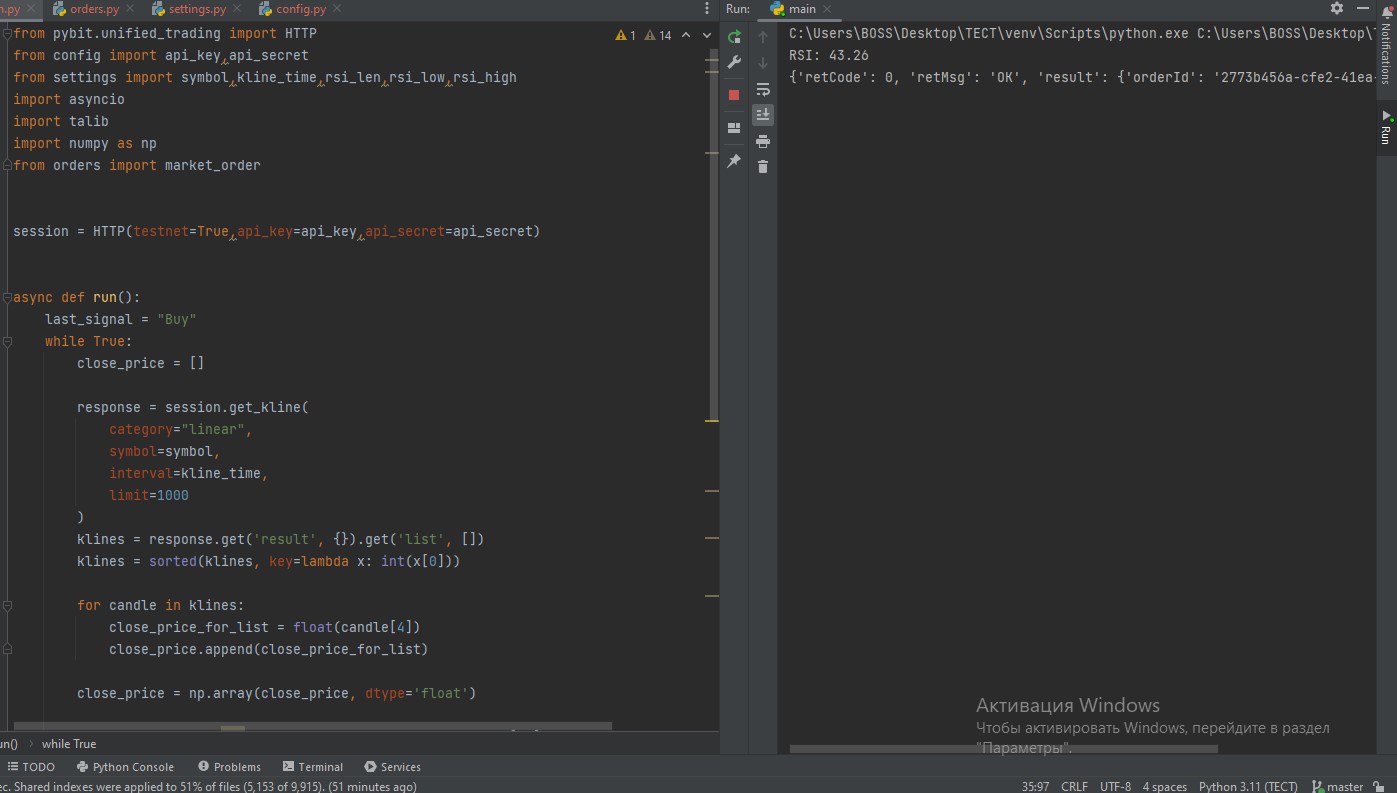
else:

qtyStep = str(float(qtyStep))

return int(len(qtyStep)),int(len(tickSize))

Итак, получили правильное количество знаков после запятой для количества и для конечной цены.

Все, теперь у нас есть ордера для закрытия и открытия сделок. Давайте это быстренько протестируем. Вместо 30 и 70 укажем какие-нибудь значения, чтобы условие срабатывало часто. Например 50 и 60. Запустим нашу программу.



Ордер размещен, все, теперь ждем, так как мы зашли в Long, теперь должны ждать сигнал на Sell. Итак, важный момент, давайте укажем, что если у нас последний сигнал был на покупку, то два раза, чтобы программа не заходила, чтобы два раза она не покупала, а чередовала покупка-продажа и наоборот.

Укажем last\_signal = None и добавим условие в нашу стратегию

if rsi\_value < rsi\_low and last\_signal != "Buy":

"""BUY"""

await market\_order(session,symbol,"Buy",close\_price)

last\_signal = "Buy"

elif rsi\_value > rsi\_high and last\_signal != "Sell":

"""SELL"""

await market\_order(session,symbol,"Sell",close\_price)

last\_signal = "Sell"

Всё, теперь у нас не будет спама ордеров и у нас будут ордера чередоваться покупка, продажа и наоборот. В принципе, это всё

# Глава 4: Разработка телеграмм-робота помощника.

# 4.1 Теоретические основы

- Телеграмм-робот (Telegram bot) - это программа, автоматизированный аккаунт, которая может взаимодействовать с пользователями через мессенджер Telegram. Он может предоставлять информацию, выполнять команды, отвечать на вопросы, отправлять уведомления и многое другое.

- Фьючерсный аккаунт (futures account) - это счет, открытый у брокера для торговли фьючерсами на товары, валюты, акции и другие активы. Фьючерс (futures) - это стандартизированный финансовый договор, предусматривающий куплю или продажу актива на определенную дату в будущем по заранее оговоренной цене. Такой аккаунт используется для спекулятивных целей, хеджирования рисков и инвестиций.

Принципы работы телеграмм-роботов основаны на API (программный интерфейс приложения) Telegram, который позволяет им взаимодействовать с мессенджером и его пользователями. Вот некоторые основные принципы работы телеграмм-роботов:

1. Регистрация и настройка: Для создания телеграмм-робота необходимо зарегистрировать его в Telegram и получить API-ключ. Далее проводится настройка функционала, сообщений, команд и других параметров робота.

2. Взаимодействие с пользователями: Телеграмм-робот может принимать сообщения от пользователей, обрабатывать их и отправлять ответы. Пользователи общаются с роботом через чаты, команды, кнопки и другие элементы интерфейса.

3. Реализация функционала: Разработчики определяют функционал робота - отображение информации, отправка уведомлений, выполнение команд, интеграция с другими сервисами и т.д. Все действия робота основаны на программировании и обработке запросов.

4. Сохранение данных и состояний: Телеграмм-роботы могут сохранять данные и состояния пользователей, чтобы предоставлять персонализированный опыт и хранить историю интеракций.

5. Обновления и поддержка: Для того чтобы робот оставался актуальным и работал без ошибок, требуется обновление софта, исправления багов, поддержка новых функций и т.д.

Основные принципы контроля фьючерсного аккаунта включают в себя следующие аспекты:

1. Мониторинг позиций: Контроль позиций открытых фьючерсных контрактов, оценка их стоимости и движения цен. Это позволяет оценить текущее финансовое положение и принимать решения о дальнейших действиях.

2. Управление рисками: Оценка рисков, связанных с фьючерсными сделками, и разработка стратегии управления рисками. Это может включать в себя установку стоп-лоссов, определение уровней стоп-аута, рассмотрение вариантов хеджирования и др.

3. Анализ рынка: Проведение анализа рынка, изучение фундаментальных и технических данных, оценка долгосрочных и краткосрочных тенденций, прогнозирование ценовых движений. Это позволяет принимать обоснованные решения при торговле на фьючерсном рынке.

4. Оперативные действия: Быстрая реакция на изменения в рыночной ситуации, мониторинг новостей и событий, которые могут повлиять на цены активов, принятие оперативных решений по закрытию или открытию позиций.

5. Торговля по стратегии: Соблюдение заранее разработанной стратегии торговли на фьючерсном рынке, установленных правил и целей, постепенное наращивание прибыли, минимизация потерь.

# 4.2 Анализ существующих решений

На сегодняшний день существует множество телеграмм-роботов помощников для торговли на онлайн крипто валютных биржах. Рассмотрим некоторые из них:

1. Cornix Bot: Cornix предлагает автоматизированные торговые стратегии для крипто валютных бирж. Робот обеспечивает управление рисками, автоматическое копирование сделок и многое другое.

2. 3Commas Bot: 3Commas позволяет создавать собственные торговые боты с различными стратегиями на крипто валютных биржах. С его помощью можно управлять портфелем, устанавливать стоп-лоссы и тейк-профиты.

3. Cryptohopper: Cryptohopper предлагает возможность создания и настройки торговых ботов для различных крипто валютных бирж. Он поддерживает автоматическое выполнение стратегий, обратное тестирование и интеграцию с TradingView.

4. Zignaly: Zignaly предлагает торговые сигналы, автоматизированные стратегии и копирование сделок на крипто валютных биржах. Робот имеет удобный интерфейс и мощные инструменты для управления портфелем.

5. Wunderbit Trading: Wunderbit Trading предлагает возможность автоматизировать торговлю на крипто валютных биржах через API. Сервис предоставляет интеграцию с Binance, BitMEX и другими популярными биржами.

Эти телеграмм-роботы помощники предоставляют удобные инструменты для автоматизации и управления торговлей на крипто валютных биржах, что помогает улучшить результаты и снизить риски для трейдеров. Важно помнить, что использование таких роботов требует осторожности и грамотного подхода для успешной торговли.

# 4.3 Разработка телеграмм-робота помощника

Робот помощник для торговли на онлайн крипто биржах обычно имеет следующие основные функции:

1. Автоматизированная торговля: Робот помощник может выполнить торговые операции автоматически на основе заранее заданных алгоритмов и стратегий торговли, что позволяет снизить риски и увеличить прибыльность сделок.

2. Мониторинг рынка: Робот постоянно отслеживает изменения цен криптовалют на бирже и предоставляет информацию о самых актуальных тенденциях и трендах на рынке.

3. Уведомления и аналитика: Робот помощник может предоставлять уведомления о важных событиях на рынке, а также аналитическую информацию, которая поможет принять правильное решение при совершении сделок.

4. Управление портфелем: Робот помощник может помочь управлять портфелем криптовалют, показывая текущую стоимость активов, предлагая оптимальное распределение средств и помогая принимать решения о дальнейших инвестициях.

5. Безопасность: Робот помощник обеспечивает безопасность средств пользователя, используя защищенные протоколы и шифрование данных при выполнении операций на бирже.

6. Поддержка 24/7: Робот помощник доступен для работы круглосуточно и готов помочь пользователям в любое время суток с выполнением операций на бирже и решением возникающих проблем.

Таким образом, робот помощник для торговли на онлайн крипто биржах предоставляет широкий спектр функций, которые помогают упростить и автоматизировать процесс торговли, увеличивая эффективность и результативность операций на рынке криптовалют.

# 4.4 Выбор библиотек и изучение API биржи для фьючерсной торговли

Существует несколько библиотек для создания телеграмм роботов на языке Python. Некоторые из наиболее популярных и широко используемых библиотек включают в себя:

1. pyTelegramBotAPI: Это популярная библиотека для работы с Telegram Bot API на языке Python. Она предоставляет простой и удобный интерфейс для создания ботов, отправки сообщений, управления командами и других функций.

2. python-telegram-bot: Еще одна популярная библиотека для создания телеграмм роботов на Python. Она также обладает широким функционалом и удобным API для работы с Telegram Bot API.

3. aiogram: Это мощная библиотека для работы с Telegram Bot API, основанная на асинхронном программировании. Она предлагает улучшенную поддержку асинхронных операций и позволяет создавать более эффективные и производительные боты.

4. Telepot: Еще одна популярная библиотека для создания телеграмм роботов на Python. Она предоставляет простой и интуитивно понятный интерфейс для работы с Telegram Bot API.

5. pyTelegramBotAPI-Wrapper: Это обертка для библиотеки pyTelegramBotAPI, которая упрощает использование API и предоставляет дополнительные функции для создания ботов на Python.

Выбор пал на библиотеку aiogram, так же в моей работе не обойтись без библиотеки CCXT (CryptoCurrency eXchange Trading Library) представляет собой мощный инструмент для работы с различными крипто валютными биржами посредством их API. Ее основное предназначение — упрощение процесса взаимодействия с биржами для выполнения торговых операций, получения информации о рынке, управления аккаунтами и анализа данных. В частности, CCXT используется для следующих целей:

1. Унификация API различных бирж: Криптовалютный мир отмечен большим разнообразием бирж, каждая из которых имеет свой уникальный API. Библиотека CCXT предоставляет унифицированный интерфейс для работы с множеством бирж, позволяя программистам использовать один и тот же код для взаимодействия с различными платформами, что значительно упрощает разработку приложений для торговли криптовалютами.

2. Получение рыночных данных: С помощью CCXT можно легко получить доступ к данным рынка в реальном времени, таким как цены, объемы торгов, стаканы заказов (order book) и т.д. Это делает библиотеку полезной для анализа рынка, разработки торговых стратегий и принятия обоснованных решений.

3. Выполнение торговых операций: Библиотека позволяет взаимодействовать с торговыми функциями бирж, такими как размещение и отмена ордеров, проверка статуса ордера, управление маржой и прочее. Это открывает возможности для создания автоматических торговых систем и ботов.

4. Управление аккаунтом: CCXT упрощает выполнение таких операций, как запрос баланса аккаунта, проверка истории торгов и транзакций, что важно для мониторинга и управления средствами.

5. Арбитраж и интеграция с криптофинансовыми сервисами: Благодаря широкому охвату бирж библиотека является отличным инструментом для арбитражных стратегий, позволяя быстро сравнивать цены и выполнять торговые операции на разных платформах. Она также используется для интеграции с различными криптовалютными кошельками, платежными системами и финансовыми приложениями.

6. Разработка образовательных и исследовательских проектов: Для изучения рынка криптовалют, тестирования и разработки новых торговых стратегий библиотека CCXT предоставляет простой доступ к рыночным данным и торговым функциям множества бирж, что может быть весьма полезно в образовательных и исследовательских целях.

Библиотека CCXT стала популярным инструментом среди разработчиков благодаря своей гибкости, поддержке большого числа бирж и простоте использования. Она активно поддерживается сообществом и регулярно обновляется, включая новые фичи и расширяя список поддерживаемых бирж.

Когда я взялся за изучение API биржи Bybit с целью создания телеграмм-робота, задача казалась весьма обширной и сложной. Bybit предоставляет мощный инструментарий для автоматической торговли, и мне хотелось максимально использовать его возможности.

Первым шагом был тщательный анализ документации API Bybit. Я обнаружил, что API поддерживает множество операций: от получения актуальных котировок до выполнения торговых операций и управления ордерами. Для реализации своего робота мне особенно интересны были методы REST API для создания и управления ордерами, а также WebSocket API для подписки на рыночные данные в реальном времени.

# Глава 5: Создание телеграмм-робота для контроля фьючерсного аккаунта

Для этого создадим новую папку и откроем ее в редакторе нашего кода.

Создадим новый файл main.py, в котором будем писать код. И сделаем виртуальное окружение командой в терминале: “python -m venv venv”. Это нужно для того, чтобы мы не захламляли наш компьютер установленными библиотеками. Когда мы создадим новое виртуальное окружение, каждая библиотека, которую мы используем в проекте, будет устанавливаться в само окружение и это не будет мешать нашей работе.

Здесь нам понадобится библиотека pybit. Собственно, она устанавливается pip install bybit. Это официальная библиотека биржи Bybit для работы с API Bybit с помощью Python. И также нам понадобится установить aiogram библиотека для работы с Telegram. Об этих библиотеках я уже говорил выше.

from config import API\_KEY, API\_SECRET, TOKEN

import ccxt

from aiogram import Bot, types

from aiogram.dispatcher import Dispatcher

from aiogram.utils import executor

from aiogram.types import ReplyKeyboardMarkup, KeyboardButton, ReplyKeyboardRemove

В Python, при разработке асинхронных ботов для мессенджера Telegram с использованием фреймворка Aiogram, инструкция from aiogram import Bot, types импортирует в наш файл кода два важных элемента: класс Bot и модуль types.

### Класс Bot

Класс Bot представляет собой центральный компонент библиотеки, который используется для взаимодействия с Telegram Bot API. Этот класс предоставляет интерфейс для отправки запросов к API Telegram. Я использую объект Bot для выполнения всех действий, которые наш бот может совершать в Telegram, например, отправлять сообщения, изображения, обрабатывать команды и т.д.

Нам необходимо создать экземпляр класса Bot и передать ему токен нашего бота, который я получил от BotFather в Telegram. Этот токен служит нашим ключом аутентификации при взаимодействии с API.

### Модуль types

Модуль types Aiogram включает в себя определения классов для различных типов данных, которые используются в Telegram Bot API. Эти классы соответствуют различным структурам данных, таким как сообщения, клавиатуры, коллбэки и множество других объектов.

Использование типизированных объектов из модуля types значительно упрощает работу с данными, позволяя эффективнее использовать автодополнение в текстовом редакторе и сокращать количество ошибок при работе с атрибутами и методами объектов.

В разработке телеграмм-ботов на Python с использованием фреймворка Aiogram, конструкция from aiogram.dispatcher import Dispatcher импортирует класс Dispatcher. Этот класс является ключевым компонентом фреймворка и выполняет роль "диспетчера" сообщений, управляя потоком данных между вашим ботом и Telegram API.

### Основные функции Dispatcher:

1. Распределение сообщений: Dispatcher анализирует входящие сообщения и направляет их к соответствующим обработчикам. Эта система позволяет разработчику определить функции (или "хэндлеры"), которые будут вызваны в ответ на различные события, например, текстовые сообщения, команды, действия с callback-кнопками и так далее.

2. Управление хэндлерами: Основной задачей Dispatcher является регистрация и управление обработчиками сообщений. Разработчик может легко добавлять, удалять или изменять приоритеты обработчиков, контролируя тем самым логику обработки сообщений и команд.

3. Асинхронная обработка событий: Поскольку Aiogram разработан для создания асинхронных приложений, Dispatcher поддерживает асинхронную обработку событий. Это означает, что ваш бот может эффективно обрабатывать множество запросов одновременно без блокировки выполнения программы.

4. Интеграция с механизмами обработки состояний: Aiogram предлагает механизмы для работы с состояниями в контексте чата или пользователя, позволяя реализовывать сложные сценарии взаимодействия. Dispatcher используется для интеграции этих механизмов в обработку сообщений и команд.

Инструкция from aiogram.utils import executor используется для импортирования модуля executor. Этот модуль содержит утилиты, необходимые для запуска и управления асинхронными задачами и циклами событий вашего бота.

### Основные функции executor:

1. Запуск бота: executor предоставляет функцию start\_polling, которая используется для запуска бота в режиме опроса (polling). В этом режиме бот постоянно опрашивает сервера Телеграм на предмет новых сообщений и обрабатывает их асинхронно.

2. Запуск вебхука: Для ботов, работающих на серверах с выходом в Интернет, может быть использована функция start\_webhook, которая позволяет боту получать уведомления о новых сообщениях через механизм вебхуков, минуя необходимость постоянного опроса.

3. Настройка: executor дает возможность настройки аспектов работы асинхронного приложения, включая логирование, грациозное завершение работы при получении сигналов остановки и так далее.

Фактически, использование executor в Aiogram является стандартным способом для запуска и управления жизненным циклом вашего асинхронного телеграм-бота. Оно позволяет абстрагироваться от низкоуровневой работы с асинхронными циклами Python, упрощая разработку и поддержку бота.

Конструкция from aiogram.types import ReplyKeyboardMarkup, KeyboardButton, ReplyKeyboardRemove импортирует классы, используемые для создания и управления клавиатурами пользователя, то есть интерактивными кнопками, которые отображаются в клиенте Telegram пользователя и упрощают процесс взаимодействия с ботом.

### Классы клавиатуры и их функции:

1. KeyboardButton:

- Этот класс представляет собой отдельную кнопку на пользовательской клавиатуре.

- Вы можете указать текст, который будет отображаться на кнопке, и другие опциональные параметры, такие как запрос контакта или геолокации пользователя непосредственно через нажатие на кнопку.

2. ReplyKeyboardMarkup:

- Этот класс используется для создания клавиатур, которые отображаются пользователю вместо стандартной клавиатуры с буквами.

- В его конструктор можно передать список списков объектов KeyboardButton, формируя тем самым необходимую структуру и внешний вид клавиатуры.

- Такие клавиатуры особенно удобны для навигации и выполнения специфичных команд в боте.

3. ReplyKeyboardRemove:

- Этот класс используется для удаления текущей пользовательской клавиатуры и возвращения к стандартной клавиатуре.

- Может быть полезен, когда взаимодействие, требующее использования пользовательской клавиатуры, завершено и клавиатура больше не нужна.

И также из модуля config импортируем токен, также там будут размещены api ключи. Затем создаем переменную bybit

bybit = ccxt.bybit({

'apiKey': API\_KEY,

'secret': API\_SECRET

})

bot = Bot(TOKEN)

print(bot)

dp = Dispatcher(bot)

которая будет обращаться к нашему USDT-фьючерсам по моим api ключам которые импортируем из модуля config.

Далее мы создаем Telegram бота, отправляем в него наш токен, который мы импортировали, и диспетчер. Подключаемся к нашему боту. Затем необходимо сделать кнопки.

b1 = KeyboardButton('/balance')

b2 = KeyboardButton('/limits')

b3 = KeyboardButton('/close\_limits')

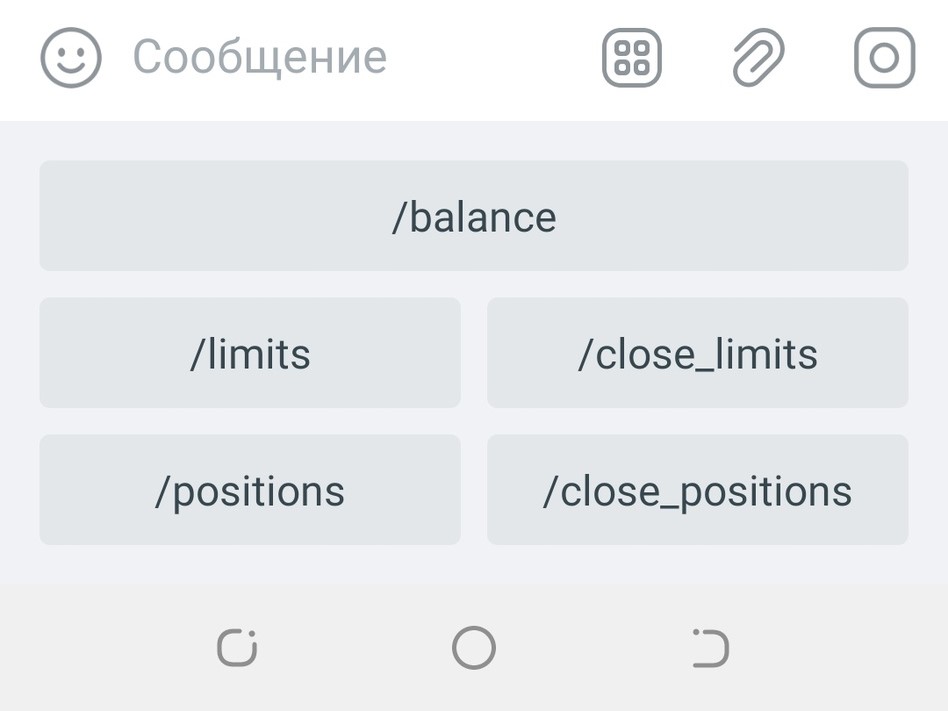
b4 = KeyboardButton('/positions')

b5 = KeyboardButton('/close\_positions')

kb\_client = ReplyKeyboardMarkup(resize\_keyboard=True, one\_time\_keyboard=True)

kb\_client.add(b1).add(b2).insert(b3).add(b4).insert(b5)

Мы импортировали с вами класс ReplyKeyboardMarkup и тут мы создаем наши кнопки. То есть у нас их будет пять, а они будут называться balance, limit и positions и соответственно закрытие этих лимитов и позиций close\_limits и close\_position . Соответственно, Balance проверяет баланс, limit - лимиты и тд. Ну и также нам нужно их подключить и расставить. Обращаемся к ReplyKeyboardMarkup и указываем размер кнопок и ставим функцию one\_time\_keyboard, то есть я хочу, чтобы кнопки убирались после нажатия. Если нужно их оставить, необходимо просто этот параметр убрать или false поставить. Далее выстраиваем кнопки, а у нас будет метод add первой, то есть с новой строчки, вторая кнопка добавится к первой строчке и третья будет с новой, то есть метод insert добавляет кнопку рядом. Если нужно, чтобы они у нас подряд были, то добавляем тоже через метод insert, таким образом делаю меню.



# 5.1 Интеграция с торговым ботом и отображение данных о фьючерсном счете

Далее создадим первый диспетчер. Диспетчер в телеграмме это функция, обработчик, который обрабатывает новые события. То есть пишется на синтаксе таким образом dp.message\_handler. Он обрабатывает команду start и команду help. То есть если пользователь ведет команду start и команду help, то, соответственно, сработает все, что мы поместим ниже в этой асинхронной функции. А что именно здесь будет? Собственно ничего. Просто мы вызовем метод sendMessage, то есть мы отправим сообщение, пользователю и напишем текст, что бот запущен.

@dp.message\_handler(commands=['start', 'help'])

async def command\_start(message: types.Message):

await bot.send\_message(message.from\_user.id, "Бот запущен", reply\_markup=kb\_client)

Далее создадим новый обработчик событий, назову его balance, то есть будет обрабатывать команду balance, получение баланса. И что будет делать функция? Я создаю переменную balance и через сессию(bybit), то есть через подключение к бирже Bybit, мы делаем запрос fetch\_balance() и вытаскиваем из всего ответа, который присылает нам Байбит, вытаскиваем ключ result, то есть собственно наш баланс.

Далее проверяем через for, перебираем все значения в переменной balance. И если walletBalance больше 0, соответственно, если у нас не нулевой баланс по какому-то активу, там USDT, BTC и прочее, то мы создадим переменную response, в которую помещаем просто наш ответ в форме строки, то есть balance пишем текстом. Далее вставляем значение, которое мы вытащили из ключа result и какие именно активы у нас на счете. Ну и на выходе await message.answer(response), то есть в ответ пользователь просто отправляем вот нашу переменную response. Это будет делать наш обработчик баланса.

@dp.message\_handler(commands=['balance'])

async def balance(message: types.Message):

balance = bybit.fetch\_balance()['info']['result']['list']

for i in balance:

if float(i['walletBalance']) > 0:

response = f"{round(float(i['walletBalance']), 2)} {i['coin']}"

await message.answer(response)

Далее сделаем обработчик лимита, создаю новую функцию limits. В ней создаю переменную open\_limits\_orders = bybit.fetch\_derivatives\_open\_orders()

Эта команда ответственна за получение открытых ордеров на деривативы с биржи Bybit. Она позволяет получить информацию о текущих открытых позициях и заказах на покупку/продажу деривативов на бирже Bybit, и далее пробегаем простым перебором для каждого элемента из этого списка мы создаем переменную дата и вытаскиваем просто имя. То есть наша задача собрать все тикеры, которые есть на фьючерсах. Через f - строку формирую ответ. То есть на выходе у нас будет список всех лимитных ордеров, которые торгуются на ByBit.

@dp.message\_handler(commands=['limits'])

async def limits(message: types.Message):

open\_limits\_orders = bybit.fetch\_derivatives\_open\_orders()

for i in open\_limits\_orders:

data = i['info']

response = f"{data['symbol']} {data['side']} Quant: {data['qty']} Price: {round( float(data['price']),2 )}"

await message.answer(response)

Следующий обработчик, закрывает все лимиты одной простой командой

@dp.message\_handler(commands=['close\_limits'])

async def close\_position(message: types.Message):

bybit.cancel\_all\_orders()

Следующий обработчик будет у нас position, следует пояснить в чем отличие позиции от лимита

В торговле на криптовалютных биржах понятия "позиция" и "лимит" имеют существенно разные значения, отражающие различные аспекты торгового процесса, разберем каждый термин отдельно.

Позиция

Позиция относится к активу (в данном случае криптовалюте), который вы держите или продаете на рынке. Позиции могут быть "длинными" (long) или "короткими" (short):

- Длинная позиция (Long Position): Это значит, что трейдер покупает криптовалюту в надежде, что ее стоимость возрастет. Трейдер войдет в длинную позицию, если ожидает роста рынка.

- Короткая позиция (Short Position): Это означает продажу криптовалюты, которой на данный момент трейдер не владеет, с намерением выкупить ее обратно позже по более низкой цене. Сюда входит заимствование криптовалюты для продажи с намерением купить ее обратно при снижении цены.

Лимит

Лимитные заказы (лимитные ордера) — это команды купить или продать криптовалюту по определенной цене или лучше. Это контраст с рыночными ордерами, которые исполняются немедленно по текущей рыночной цене. Лимитные ордера используются в следующих целях:

- Лимитный ордер на покупку: Вы устанавливаете максимальную цену, по которой готовы купить криптовалюту. Такой ордер будет исполнен только в случае, если цена криптовалюты упадет до указанного вами уровня или ниже.

- Лимитный ордер на продажу: Вы устанавливаете минимальную цену, по которой хотите продать криптовалюту. Если рыночная цена достигнет вашего указанного уровня или превысит его, ордер будет исполнен.

Резюме различий

- Позиция относится к вашему текущему участию на рынке, то есть к криптовалюте, которую вы уже купили или продали (и еще не закрыли сделку).

- Лимит является инструкцией по исполнению будущей сделки по конкретной цене или лучше, что позволяет контролировать условия входа или выхода из рыночной позиции.

Обои эти понятия являются фундаментальной частью стратегии торговли и управления рисками на криптовалютных и других финансовых рынках.

Функция у нас называется position и далее ставим переменную позицию, также обращаемся к сессии bybit через уже известную команду fetch\_derivatives\_open\_orders(), вытаскиваем результат и далее через for тоже самое перебираем каждый элемент из списка. Формирую ответ, создаю переменную response, передаю в неё символ, сторону(Bye/Sell) и объем. И на выходе распечатаем то сообщение, которое получает пользователь.

@dp.message\_handler(commands=['positions'])

async def position(message: types.Message):

positions = bybit.fetch\_derivatives\_positions()

for i in positions:

data = i['info']

response = f"{data['symbol']} {data['side']} Quant: {data['size']}"

await message.answer(response)

Последний обработчик будет закрывать все позиции, по аналогии с обработчиком positions добываем данные и меняем сторону(side) на продажу - Sell

@dp.message\_handler(commands=['close\_positions'])

async def close\_position(message: types.Message):

positions = bybit.fetch\_derivatives\_positions()

for i in positions:

data = i['info']

symbol = data['symbol']

side = data['side']

size = data['size']

if side == 'Buy':

side = 'Sell'

else:

side = 'Buy'

order = bybit.create\_market\_order(symbol, side, size)

await bot.send\_message(message.from\_user.id, "Позиции закрыты", reply\_markup=kb\_client)

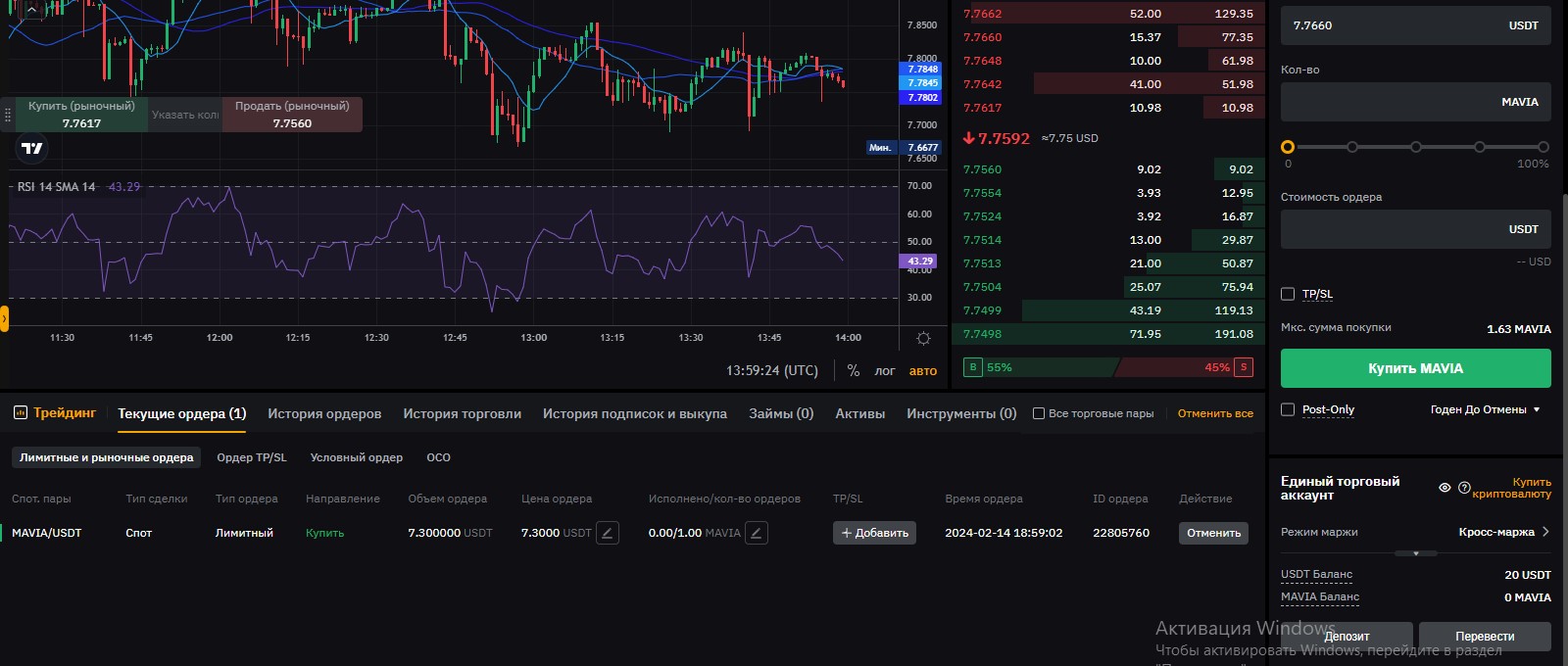
# 5.2 Тестирование и оптимизация робота для контроля фьючерсного аккаунта

Давайте посмотрим, как работает бот. У меня запущен код и в самом телеграмме мы запускаем нашего бота командой /start. Видим, что у нас есть пять кнопок. Можем нажать посмотреть баланс. Вот на этом счете у меня там 20 долларов. Давайте откроем на самом байбите



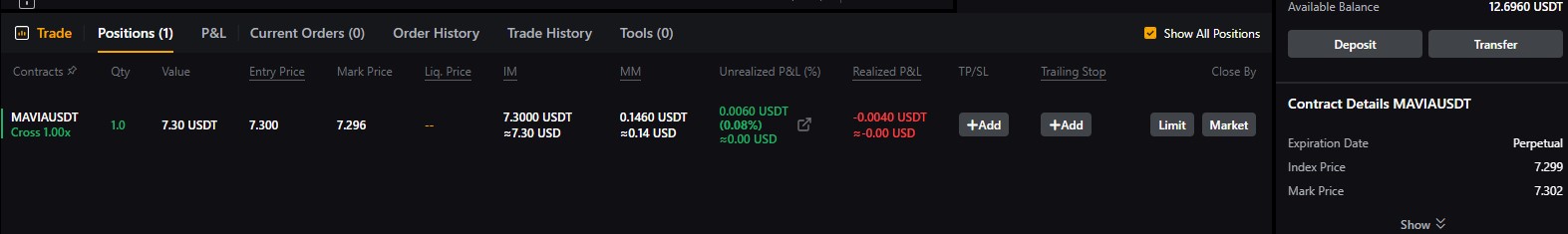


Всё хорошо, давайте дальше поставлю лимитный ордер на одну монету MAVIA по цене 7.3$ она явно меньше рыночный, значит он у нас повисит какое то время





Замечательно, купим что нибудь в лонг





Закрывающие кнопки действительно всё закрывают, а в истории отображаются как закрытые ордера.

# Глава 6: Заключение

# 6.1 Выводы по исследованию

В рамках дипломной работы была выполнена значительная работа по разработке и тестированию торгового робота для криптовалютной биржи Bybit, а также созданию телеграмм-помощника, предоставляющего пользователю актуальную информацию о балансе, активных ордерах и лимитах. Были применены современные технологии и методы программирования, а также глубоко изучены особенности работы с API крипто валютной биржи Bybit.

Стоит отметить, что хоть всё и работает, но этого недостаточно и несет в себе большой риск. Рынок имеет большую волатильность и для создания действительно хорошего и успешного робота необходимо обладать очень большими познаниями и опытом в сфере экономики это в первую очередь, и лишь только потом оформлять это кодом.

В ходе работы были достигнуты следующие результаты:

1. Разработан торговый робот, способный автоматизировать процессы покупки и продажи криптовалют на бирже Bybit. Торговая стратегия робота основана на техническом анализе рынка и эффективном распределении капитала для максимизации прибыли и минимизации рисков.

2. Создан телеграмм-бот, который обеспечивает оперативное взаимодействие с пользователем. Бот предоставляет актуальную информацию о состоянии счета, текущих ордерах и установленных лимитах. Эта функциональность значительно повышает удобство использования торгового робота и позволяет оперативно реагировать на изменения рынка.

3. Проведено всестороннее тестирование разработанных систем. В ходе тестирования были выявлены и устранены недостатки в работе торгового робота и телеграмм-бота, что в конечном итоге обеспечило высокую стабильность и надежность работы системы в целом.

4. Проанализирована эффективность использования торгового робота на реальных рыночных условиях. Анализ показал, что благодаря точно настроенной стратегии и оперативному мониторингу рынка через телеграмм-бота, торговый робот демонстрирует положительную доходность, что подтверждает его практическую ценность и эффективность вложенных усилий.

Разработка торгового робота и телеграмм-бота для работы на крипто валютной бирже Bybit является актуальной задачей, учитывая растущий интерес к торговле цифровыми активами и постоянно увеличивающиеся объёмы торгов на рынке криптовалют. Реализация данного проекта показала, что современные технологии и методы программирования позволяют создать эффективные инструменты для автоматизации торговых операций и управления рисками.

# 6.2 Перспективы дальнейших исследований в данной области

Автоматизированная криптовалютная торговля произвела революцию в способе, с помощью которого трейдеры и инвесторы взаимодействуют с рынками. Благодаря прогрессу в области алгоритмической торговли и искусственного интеллекта, предстоящие исследования обещают еще больше расширить возможности и эффективность торговых стратегий. Ниже приведены ключевые направления для будущих исследований в области автоматизированной криптовалютной торговли:

1. Развитие алгоритмов машинного обучения

Применение алгоритмов машинного обучения и глубокого обучения для предсказания движения рынка криптовалют представляет собой одно из самых многообещающих направлений. Исследования в этой области могут сосредоточиться на разработке новых моделей прогнозирования, которые способны адаптироваться к высокой волатильности и непредсказуемости крипторынков.

2. Изучение сигналов социальных медиа и новостных ресурсов

Анализ данных из социальных сетей и новостных ресурсов с помощью обработки естественного языка (NLP) может предоставить новые инсайты для торговых стратегий. Исследования могли бы фокусироваться на выявлении корреляции между сигналами социальных медиа, новостными событиями и движением рынка криптовалют.

3. Улучшение методов управления рисками

Управление рисками остается критически важной составляющей успешной торговли. Дальнейшие исследования могут исследовать новые методы для минимизации потерь, включая оптимизацию распределения портфеля, страхование ставок и разработку стратегий хеджирования для криптовалют.

4. Разработка децентрализованных торговых платформ

Децентрализованные финансы (DeFi) продолжают набирать популярность, предлагая альтернативные способы торговли и инвестирования без посредников. Изучение и разработка децентрализованных торговых ботов и платформ может стать следующим шагом в эволюции криптовалютных рынков.

5. Исследование регуляторных и этических вопросов

По мере развития рынка криптовалют растет и внимание регуляторов. Важным направлением исследований станет изучение влияния регуляторных изменений на торговлю, а также разработка этических норм и стандартов в области автоматизированной торговли.

6. Оптимизация существующих инфраструктур

Исследования нацелены на улучшение производительности торговых систем через сокращение времени задержки, повышение надежности и создание более эффективных алгоритмов обработки заказов. Это включает в себя работу как с программным обеспечением, так и с аппаратным обеспечением.

Заключение. Отрасль автоматизированной криптовалютной торговли находится на ранних этапах своего развития, и потенциальные возможности для исследований обширны. Сочетание новых технологий, таких как машинное обучение, обработка естественного языка и блокчейн, с непрерывно развивающимся рынком криптовалют создает почву для инноваций и прогресса в данной области.