Пошаговое руководство по созданию торгового бота на любом языке программирования

[Блог компании RUVDS.com](https://habr.com/ru/company/ruvds/blog/)[Программирование](https://habr.com/ru/hub/programming/)[Финансы в IT](https://habr.com/ru/hub/finance/)

Авторизованный Перевод

[Автор оригинала: Yakko Majuri](https://blog.usejournal.com/a-step-by-step-guide-to-building-a-trading-bot-in-any-programming-language-d202ffe91569)

У меня пока нет достаточной квалификации, позволяющей давать советы, касающиеся инвестиций, законов, или чего-то подобного. Я и не пытаюсь этого делать. Цель руководства заключается в том, чтобы, в учебных целях, рассказать о программировании торговых ботов на языке C#, и Python ML. Вы должны понимать, что, создав бота, вы несёте ответственность за принимаемые им решения, за выполняемые им инвестиционные операции, за те риски, которые сопряжены с торговой деятельностью. Я не могу отвечать за те решения, которые вы примете после прочтения данного материала. Помните, что некоторые боты способны потерять большие деньги, поэтому используйте их с осторожностью.  
  
Часто можно встретить статьи, которые называются примерно так: «10 проектов, которые нужно реализовать программисту». Часто в списки этих статей входят торговые боты. Я считаю, что разработка торгового бота — это достойное вложение сил. Поэтому я решил уделить некоторое время тому, чтобы написать учебное руководство об этом.  
  
Но вместо того, чтобы устраивать тут построчный разбор некоего кода, я решил, что лучше будет разобрать те концепции, с которыми должен быть знаком тот, кто хочет создать собственного бота. Смысл моего материала заключается в том, чтобы, прочтя его, вы сами написали бы код.  
  
Поэтом я расскажу о том, что полезно знать для разработки торгового бота, и о том, что для этого нужно (от работы с биржами до реализации простой торговой стратегии). Здесь же я коснусь вопросов, связанных с архитектурой и внутренним устройством простых торговых ботов, с идеями, которые лежат в их основе.  
  
[[](https://habr.com/ru/company/ruvds/blog/517234/)](https://habr.com/ru/company/ruvds/blog/517234/)  
Я буду демонстрировать примеры, написанные на псевдокоде. Поэтому вы сможете читать это руководство и тут же писать собственного бота на выбранном вами языке программирования.  
  
В результате вам будет комфортно, так как вы будете пользоваться инструментом, с которым вы хорошо знакомы. Сможете спокойно заниматься программированием, а не тратить время на настройку рабочей среды и на привыкание к новому языку.  
  
Вы выберете оружие, а я научу вас владеть этим оружием.

Шаг 1. Выбираем оружие

На первом шаге этого руководства вы выберете язык программирования, которым будете пользоваться. Этот вопрос вы должны решить самостоятельно.  
  
Некоторым языкам, вроде Python, можно отдать предпочтение в том случае, если вы планируете в будущем оснастить своего бота, например, механизмами машинного обучения. Но моя основная идея заключается в том, что вы можете выбрать тот язык, с которым вам удобнее всего работать.

Шаг 2. Ищем поле битвы

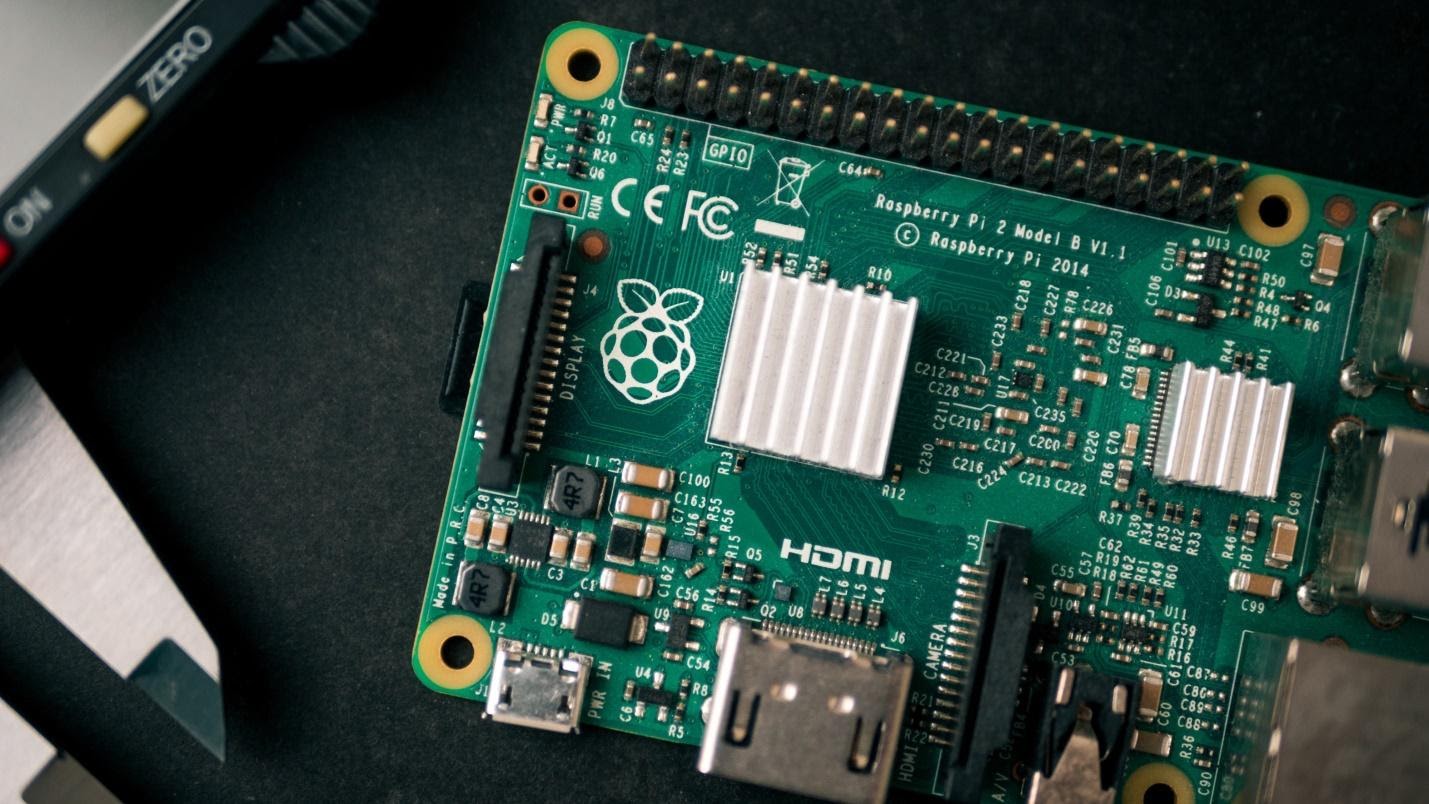


В руководствах по написанию торговых ботов часто упускают одну важную тему. Она касается выбора биржи. Дело в том, что для того чтобы бот смог бы заниматься тем, для чего он создан, ему нужен доступ к бирже, на которой можно чем-то торговать. Выбор биржи и умение ей пользоваться — это так же важно, как навыки программирования.  
  
Итак, вашим первым шагом будет принятие решения о том, чем именно вы собираетесь торговать (акциями, валютами, криптовалютами), и решения о том, где именно вы будете торговать.  
  
Если говорить о биржевых активах, то я посоветовал бы обратить внимание на криптовалюты. Причина такой рекомендации заключается не в том, что я являюсь сторонником блокчейн-технологий и криптовалют (я полностью открыт в этом вопросе), а всего лишь в том, что криптовалютные рынки работают круглые сутки 7 дней в неделю.  
  
Более «традиционными» активами можно торговать только в определённые временные промежутки, и часто — только по будним дням. Рынки акций, например, обычно открыты с 9 утра до 4 вечера и по выходным они не работают. Рынки FOREX, хотя и могут работать круглосуточно, обычно закрыты в выходные.  
  
В связи с этим нам лучше всего подойдут криптовалюты, так как выбор данного рынка позволит нашем боту работать без перерывов. Кроме того, криптовалюты известны высоким уровнем волатильности. А это значит, во-первых, то, что на торговле ими можно потерять большие деньги, и во-вторых, то, что они представляют собой отличный инструмент для изучения и тестирования торговых стратегий.  
  
О биржевых активах мы уже поговорили. Давайте теперь обсудим требования, которые нужно учитывать при выборе биржи, для которой будет создаваться бот. А именно, я говорю о двух основных требованиях:

1. У вас должна быть законная возможность торговать на выбранной бирже и работать с предлагаемыми ей торговыми инструментами. Если говорить о криптовалютах, то в некоторых странах торговля ими запрещена. Учитывайте это, выбирая инструменты и биржу.
2. Биржа должна обладать API, который доступен её клиентам. Нельзя создать торгового бота, который не отправляет запросы к бирже и не получает от неё ответов.

После того, как оказалось, что некая биржа удовлетворяет этим двум основным требованиям, вы можете проанализировать её глубже. Например, оценить размер комиссий, оценить её надёжность и известность, взглянуть на качество документации к её API.  
  
И ещё, что так же важно, как и всё остальное, я порекомендовал бы оценить объём торгов биржи. Биржи с низкими объёмами имеют свойство «отставать» от ценовых движений. На них, кроме того, сложнее бывает выполнять лимитные заявки (подробнее об этом мы поговорим ниже).  
  
Если вы, в итоге, решили выбрать криптовалюты, то [вот](https://www.coingecko.com/en/exchanges) — хороший список ведущих бирж. Здесь вы можете найти различные сведения о них, которые помогут вам в выборе биржи.

Шаг 3. Строим лагерь



Если биржа — это поле битвы, то теперь мы поговорим о том месте, из которого будем отправлять на поле битвы свои войска. Кстати, мне, пожалуй, пора завязывать с такими аналогиями.  
  
Здесь я говорю о сервере. Код бота должен выполняться на некоем сервере, что позволит боту отправлять запросы к API биржи.  
  
Во время тестирования бота, естественно, в роли сервера может выступать ваш компьютер. Но если вам нужно, чтобы бот работал бы постоянно, обычный компьютер — это, определённо, не лучший выбор.  
  
Тут у меня есть два предложения:

1. Роль сервера может выполнять Raspberry Pi (этот подход интереснее).
2. Сервером может быть некая облачная служба (а этот подход лучше).

Я так думаю, что организация деятельности бота на базе собственного Raspberry Pi-сервера — это интересная и современная идея, поэтому, если и вам эта идея нравится, вы можете претворить её в жизнь.  
  
Но большинство создателей ботов, вероятно, остановят свой выбор на каком-нибудь провайдере облачных услуг вроде [AWS](https://aws.amazon.com/), [Azure](https://azure.microsoft.com/en-gb/), [GCS](https://cloud.google.com/) или [Digital Ocean](https://www.digitalocean.com/).  
  
Большинство крупных облачных провайдеров имеют хорошие бесплатные тарифные планы, поэтому вам, возможно, удастся хостить своего бота у такого провайдера бесплатно.  
  
На этом я разговор о серверах завершаю. Вам стоит выбрать то, что лучше всего вам подходит. Для маленького проекта, вроде того, которым занимаемся мы, то, что именно будет выбрано, не особенно сильно повлияет на конечный результат работы.

Шаг 4. Создаём бота



А вот теперь начинается самое интересное. Но, прежде чем приступать к этой части нашего проекта, проверьте, выполнено ли следующее:

1. Вы зарегистрировались на бирже и получили необходимые разрешения на работу с ней.
2. У вас есть возможность работать с API биржи, у вас имеется ключ API.
3. Вы выбрали хостинг для бота.

Если эти вопросы решены, это значит, что мы можем двигаться дальше.

▍Простейший бот

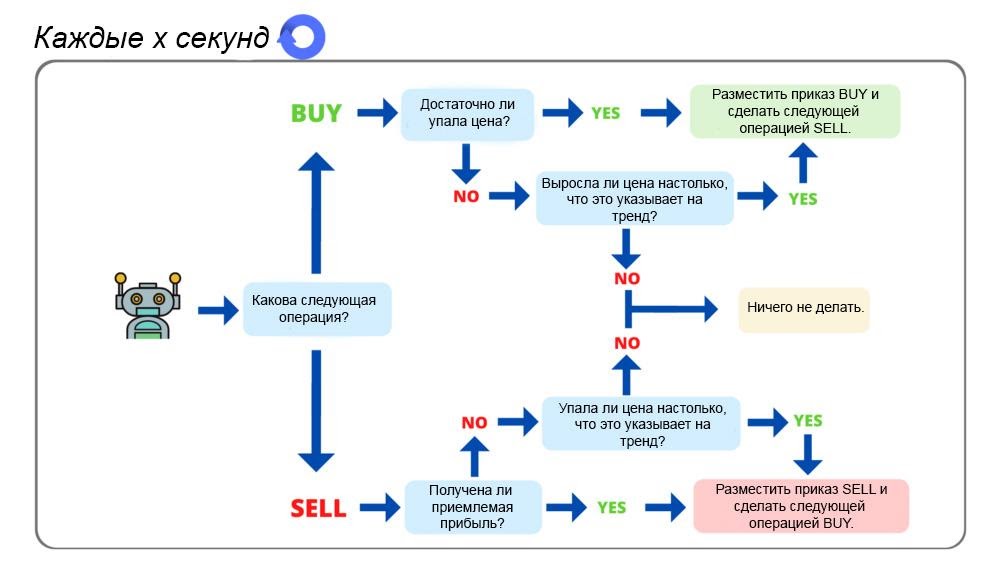
Моя цель заключается в том, чтобы помочь тому, кто до этого момента совершенно ничего не знал о ботах, подняться до уровня создания простого действующего бота. Поэтому я расскажу о том, как создать простого торгового бота, которого вы сможете, в соответствии с вашими нуждами, расширять и улучшать.  
  
У нашего бота будут некоторые ограничения:

1. Бот сможет пребывать лишь в одном из двух состояний: BUY (покупка) или SELL (продажа). Он не будет постоянно размещать заявки на покупку или на продажу по разным ценам. Если последней операцией была продажа, то следующей операцией, которую попытается выполнить бот, будет покупка.
2. Бот будет использовать фиксированные пороговые значения для принятия решений о покупке и продаже. Более интеллектуальный бот может быть способен самостоятельно настраивать подобные значения, основываясь на различных индикаторах, но стратегия и ограничения нашего бота будут задаваться вручную.
3. Он будет торговать только одной валютной парой. Например — BTC/USD.

Эти ограничения упрощают нашу задачу. Бот будет простым, а значит, его легче будет создать и поддерживать. Это же позволит нам очень быстро развёртывать его код на сервере. В целом, тут мы говорим о принципе KISS.

▍Механизм принятия решений

Вот простая диаграмма, дающая общий обзор функционирования нашего бота.

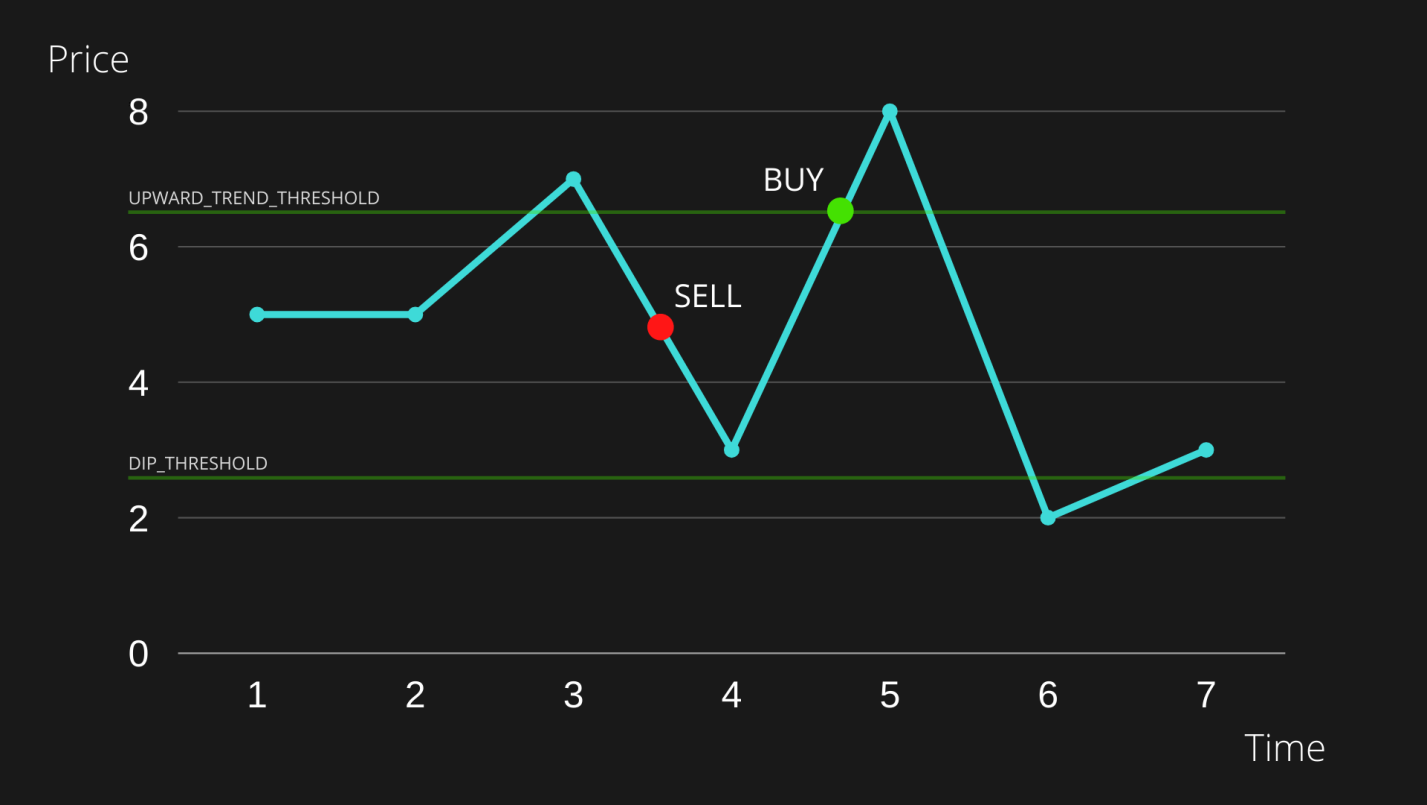


А теперь можно приступать к планированию архитектуры бота.  
  
Нам, для начала, понадобится переменная, в которой будут храниться сведения о том, в каком именно состоянии находится бот в текущий момент. Это либо BUY, либо — SELL. Для хранения подобных сведений хорошо подойдёт логическая переменная или перечисление.  
  
Затем нужно задать пороговые значения для операций покупки и продажи. Эти значения выражаются в процентах, они представляют собой увеличение или уменьшение цены актива с момента выполнения предыдущей операции.  
  
Например, если я купил что-то по цене в $100, а в настоящий момент цена составляет $102, то мы имеем дело с увеличением цены на 2%. Если порог для операции SELL установлен на однопроцентное увеличение цены, то бот, увидев эти 2%, продаст актив, так как он уже получил прибыль, превышающую заданное нами пороговое значение.  
  
В нашем случае подобные значения будут константами. Нам понадобится 4 таких значения — по 2 на каждое состояние бота.

▍Пороговые значения для выполнения операции BUY (если бот находится в состоянии SELL)

* DIP\_THRESHOLD: бот выполняет операцию покупки в том случае, если цена уменьшилась на значение, большее, чем задано DIP\_THRESHOLD. Смысл этого заключается в реализации стратегии «покупай дёшево, продавай дорого». То есть, бот будет пытаться купить актив по заниженной цене, ожидая роста цены и возможности выгодной продажи актива.
* UPWARD\_TREND\_THRESHOLD: бот покупает актив в том случае, если цена выросла на значение, превышающее то, что задано этой константой. Этот ход противоречит философии «покупай дёшево, продавай дорого». Его цель заключается в том, чтобы выявить восходящий тренд и не пропустить возможность покупки до ещё большего роста цены.

Вот иллюстрация, которая может помочь в понимании смысла этих констант.



Если мы выполнили операцию SELL в момент, отмеченный на рисунке красным маркером SELL, то после этого бот, принимая решение о выполнении операции BUY, будет руководствоваться пороговыми значениями DIP\_THRESHOLD и UPWARD\_TREND\_THRESHOLD.  
  
Если цена уйдёт ниже нижней зелёной линии или выше верхней зелёной линии, мы выполним операцию BUY. В ситуации, показанной на рисунке, цена ушла выше верхнего предела. Поэтому мы, руководствуясь значением UPWARD\_TREND\_THRESHOLD, выполнили операцию BUY.

▍Пороговые значения для выполнения операции SELL (если бот находится в состоянии BUY)

* PROFIT\_THRESHOLD: бот продаёт актив в том случае, если цена стала выше цены, вычисленной на основе этого значения, так как ранее актив был куплен по более низкой цене. Именно так мы получаем прибыль. Мы продаём актив по цене, которая выше той, что была в момент его покупки.
* STOP\_LOSS\_THRESHOLD: в идеальной ситуации мы хотели бы, чтобы бот продавал бы активы только тогда, когда продажа приносит нам прибыль. Но, возможно, произошло сильное движение рынка вниз. В такой ситуации мы решим выйти из сделки до того, как понесём слишком большие убытки, и позже купить актив по более низкой цене. Это пороговое значение используется для закрытия позиции с убытком. Цель этой операции — предотвращение более сильных потерь.

Вот иллюстрация.



Тут показана ситуация, когда там, где стоит маркер BUY, была сделана покупка. После этого цена достигает предела, заданного PROFIT\_THRESHOLD, и мы продаём актив с прибылью. Именно так боты зарабатывают.  
  
Теперь, после того, как у нас сформировалось общее понимание того, как функционирует бот, пришло время рассмотреть псевдокод.

▍Вспомогательные функции для работы с API

Первое, что понадобится боту, это возможности обмена данными с API биржи. Эти возможности мы реализуем с помощью нескольких вспомогательных функций:

FUNCTION getBalances():

    DO: Выполнить GET-запрос к API биржи для получения

    сведений о балансах

    RETURN: Сведения о балансах

FUNCTION getMarketPrices():

    DO: Выполнить GET-запрос к API биржи для получение

    текущей цены актива

    RETURN: Текущая цена актива

FUNCTION placeSellOrder():

    DO:

        1. Вычислить количество актива для продажи (на основе

        некоего заданного порогового значения, например,

        50% общего баланса)

        2. Отправить POST-запрос к API биржи для выполнения

        операции SELL

    RETURN: Цена совершения сделки

FUNCTION placeBuyOrder():

    DO:

        1. Вычислить количество актива для покупки (на основе

        некоего заданного порогового значения, например,

        50% общего баланса)

        2. Отправить POST-запрос к API биржи для выполнения

        операции BUY

    RETURN: Цена совершения сделки

// Необязательная функция, которая предназначена для

// получения подтверждения выполнения операций

FUNCTION getOperationDetails():

    DO: Выполнить GET-запрос к API биржи для получения

    сведений об операции

    RETURN: Сведения об операции

Вышеприведённый код должен быть понятен без особых объяснений. Но, реализуя этот функционал, вы должны точно знать о том, какие именно данные нужно передать API при выполнении POST-запросов, инициирующих покупку или продажу чего-либо.  
  
Часто, например, когда торгуют парой XAU/USD (золото и доллары США), при выполнении запроса можно указать или то, сколько золота нужно купить, или то, сколько долларов нужно продать. При выполнении подобных запросов очень важно чётко понимать смысл производимых действий.

▍Главный цикл бота

Теперь, когда мы подготовили вспомогательные функции, приступим к описанию действий бота. Для начала нам нужно создать бесконечный цикл, выполняемый с некоторой периодичностью. Предположим, нам нужно, чтобы бот пытался бы выполнить некую операцию каждые 30 секунд. Вот как может выглядеть такой цикл:

FUNCTION startBot():

    INFINITE LOOP:

        attemptToMakeTrade()

        sleep(30 seconds)

Далее — настроим переменные и константы, о которых мы говорили выше, и напишем логику бота, позволяющую ему принимать решения. В результате, помимо вспомогательных функций и главного цикла, основной код бота будет выглядеть так:

bool isNextOperationBuy = True

const UPWARD\_TREND\_THRESHOLD = 1.50

const DIP\_THRESHOLD = -2.25

const PROFIT\_THRESHOLD = 1.25

const STOP\_LOSS\_THRESHOLD = -2.00

float lastOpPrice = 100.00

FUNCTION attemptToMakeTrade():

    float currentPrice = getMarketPrice()

    float percentageDiff = (currentPrice - lastOpPrice)/lastOpPrice\*100

    IF isNextOperationBuy:

        tryToBuy(percentageDiff)

    ELSE:

        tryToSell(percentageDiff)

FUNCTION tryToBuy(float percentageDiff):

    IF percentageDiff >= UPWARD\_TREND\_THRESHOLD OR percentageDiff <= DIP\_THRESHOLD:

        lastOpPrice = placeBuyOrder()

        isNextOperationBuy = False

FUNCTION tryToSell(float percentageDiff):

    IF percentageDiff >= PROFIT\_THRESHOLD OR percentageDiff <= STOP\_LOSS\_THRESHOLD:

        lastOpPrice = placeSellOrder()

        isNextOperationBuy = True

Обратите внимание на то, что пороговые значения в этом коде выбраны произвольным образом. Вам следует подобрать эти значения самостоятельно, в соответствии с применяемой вами торговой стратегией.  
  
Если вышеприведённый код совместить со вспомогательными функциями и с главным циклом бота, который может быть представлен чем-то вроде функции main, это будет означать, что теперь в нашем распоряжении имеется очень простой работающий бот, обладающий основными возможностями, характерными для ботов.  
  
На каждой итерации цикла бот будет проверять своё текущее состояние (BUY или SELL) и будет пытаться выполнить торговую операцию, используя при анализе текущей ситуации жёстко заданные в его коде пороговые значения. Затем, если операция будет выполнена, бот обновит сведения о своём текущем состоянии и данные о цене, по которой была совершена последняя операция.

Шаг 5. Улучшаем бота



Базовая архитектура нашего бота готова. Но мы, вероятно, можем его немного улучшить, оснастив его некоторыми дополнительными возможностями.

▍Журналы

Когда я приступил к созданию одного из вариантов этого бота, для меня было очень важно то, чтобы бот постоянно бы логировал сведения о своих действиях, выводя их в терминал и в отдельный файл журнала.  
  
Каждый шаг работы программы должен был сопровождаться примерно такими записями:

[BALANCE] USD Balance = 22.15$

[BUY] Bought 0.002 BTC for 22.15 USD

[PRICE] Last Operation Price updated to 11,171.40 (BTC/USD)

[ERROR] Could not perform SELL operation - Insufficient balance

То, что попадает в файл журнала, снабжается отметками времени. В результате, если я подключаюсь к серверу раз в день, и вижу, например, сообщение об ошибке, я могу узнать о том, когда именно произошла ошибка, и о том, чем бот занимался всё это время.  
  
Оснастить бота подобными возможностями — значит написать функцию, которую можно назвать createLog. Эта функция должна вызываться на каждом шаге главного цикла бота. Вот как может выглядеть эта функция:

FUNCTION createLog(string msg):

    DO:

        1. Вывести msg в терминал

        2. Записать msg в файл журнала, добавив отметку времени

▍Идентификация трендов

Главная цель нашего бота заключается в том, чтобы дёшево покупать активы и продавать их, получая прибыль. Но в его коде есть две константы, символизирующие два пороговых значения, которые отчасти этой цели противоречат. Это UPWARD\_TREND\_THRESHOLD и STOP\_LOSS\_THRESHOLD.  
  
Эти значения предназначены для ограничения убытков путём продажи актива при падении цены и для организации покупки актива при росте цены. Смысл тут в том, что с их помощью мы пытаемся предугадать тренды, цены при появлении которых выходят за пределы обычной стратегии, но могут либо нам навредить, либо дать нам заработать. И то и другое предусматривает некие действия с нашей стороны.  
  
То, как сейчас устроен анализ цены, очень сильно нас ограничивает. Анализ цены, проводимый путём сравнения пары показателей, далёк от механизма, позволяющего выявлять тренды.  
  
Но мы, к счастью, можем, без особенных проблем, сделать нашу систему определения трендов более надёжной.  
  
Нам нужно лишь организовать наблюдение за большим количеством ценовых значений, чем раньше. А раньше мы хранили сведения лишь об одном ценовом показателе — о стоимости актива на момент последней операции (lastOpPrice). Можно, например, хранить сведения о ценах за 10 или 20 последних итераций цикла бота и сравнивать с текущей ценой их, а не только lastOpPrice. Это, вероятно, позволит лучше идентифицировать тренды, так как при таком подходе мы можем уловить краткосрочные колебания цены, а не колебания, происходящие за долгое время.

▍База данных?

Простому боту, на самом деле, база данных не нужна. Ведь он оперирует весьма небольшими объёмами данных и хранит всю необходимую ему информацию в памяти.  
  
Но что произойдёт в том случае, если, например, бот будет аварийно остановлен? Как ему узнать, без вмешательства человека, о том, каким было значение lastOpPrice?  
  
Для того чтобы исключить необходимость ручного вмешательства в код бота при его перезапуске нам может понадобиться некая простая база данных, в которой можно хранить какие-то показатели вроде lastOpPrice.  
  
При таком подходе бот, запускаясь, может не использовать значения, жёстко заданные в коде. Вместо этого он обращается к сохранённым данным и продолжает работу с того места, где она была прервана.  
  
В зависимости от того, насколько простой, по вашему мнению, должна быть эта «база данных», вы можете даже решить использовать в таком качестве обычные .txt- или .json-файлы, так как, в любом случае, речь идёт о хранении весьма ограниченного набора данных.

▍Панель управления



Если вы хотите организовать визуализацию деятельности бота, а так же управлять им, не редактируя его код, то вам может захотеться подключить бота к некоей панели управления.  
  
Это потребует наличия у бота собственного серверного API, предназначенного для управления его функционалом.  
  
Подобный подход, например, позволит легко менять пороговые значения.  
  
Существует множество шаблонов панелей управления, а значит вам, если вы решите сделать что-то подобное, даже не придётся создавать такую панель с нуля. Взгляните, например, на [Start Bootstrap](https://startbootstrap.com/themes/) и [Creative Tim](https://www.creative-tim.com/).

▍Тестирование стратегий на исторических данных

Многие биржи дают клиентам доступ к историческим ценовым данным. Кроме того, если вам нужны такие данные, их обычно несложно достать.  
  
Их использование весьма полезно для тестирования торговых стратегий перед их реальным применением. Это позволяет запустить симуляцию, используя исторические данные и «ненастоящие» деньги. Благодаря этому можно узнать о том, насколько удачно показали бы себя пороговые значения, и, если нужно, поменять эти значения.

▍Дополнительные сведения о пороговых значениях и заявках

При размещении заявок нужно учитывать несколько моментов.  
  
Во-первых, нужно знать о том, что существуют два типа заявок: лимитные и рыночные. Если вы совсем ничего об этом не знаете — вам, определённо, стоит почитать специальную литературу. Я тут объясню эти идеи буквально в двух словах.  
  
Рыночные заявки — это заявки, которые исполняются по текущей рыночной цене. В большинстве случаев это означает их немедленное исполнение.  
  
Лимитные заявки, с другой стороны, это заявки, которые размещают, указывая цену, которая ниже рынка (в случае с заявками на покупку), или выше рынка (в случае с заявками на продажу). При этом нет гарантии того, что эти заявки будут исполнены, так как цена может не достигнуть заданного значения.  
  
Сильная сторона лимитных заявок заключается в том, что они позволяют, предугадывая движения рынка, размещать заявки там, где, по мнению трейдера, может оказаться цена.  
  
Кроме того, к таким заявкам обычно применяются более низкие комиссии, чем к рыночным. Это так из-за того, что к рыночным заявкам обычно применимо то, что называется «taker fee» («комиссия получателя»), а к лимитным заявкам — то, что обычно называется «maker fee» («комиссия создателя»).  
  
Причины, по которым эти комиссии называются именно так, заключаются в том, что тот, кто размещает рыночную заявку, просто принимает («taking») текущую рыночную цену. А лимитные заявки находятся за пределами рыночных цен, они добавляют рынку ликвидности и, в результате, «создают рынок», за что их создатели вознаграждаются более низкими комиссиями.  
  
Обратите внимание на то, что бот, который мы тут рассматриваем, лучше всего подходит для работы с рыночными заявками.  
  
И, завершая разговор о комиссиях, хочу отметить, что задавая значение PROFIT\_THRESHOLD нужно учитывать и комиссии.  
  
Для того чтобы получить прибыль, бот должен сначала выполнить операцию BUY, а потом — операцию SELL. А это значит, что комиссия будет взята два раза.  
  
В результате бота нужно настроить так, чтобы прибыль, получаемая с продаж, по меньшей мере, покрывала бы комиссии. В противном случае бот будет торговать в убыток.  
  
Поразмыслим об этом, исходя из предположения о применении комиссий, не зависящих от суммы заявки и от вида операции. Итак, комиссия за покупку актива на $100,00 составляет $0,50. Если этот актив будет продан за $100,75 и при этом будет взята такая же комиссия, то окажется, что валовая прибыль составляет 0,75%. Но, на самом деле, тут мы имеем дело с чистым убытком в 0,25%.  
  
А теперь представьте себе, что ваш бот всегда закрывает сделки с чистым убытком. В такой ситуации можно довольно быстро потерять немалые деньги.

Итоги

Моей основной целью было раскрытие концепций, о которых нужно знать при разработке торговых ботов. Я старался описать всё так, чтобы это было понятно даже тем, кто раньше никогда не торговал на бирже. При этом я не привязывал повествование к какому-то конкретному языку программирования.  
  
Я исходил из предположения о том, что читатели этого материала знают о том, как, пользуясь выбранным ими языком программирования, выполнять HTTP-запросы. Поэтому в детали программирования я не вдавался, сосредоточившись на других вещах.  
  
Я рассказал вам о разработке торговых ботов всё, что хотел. Надеюсь, теперь вы сможете создать собственного бота.  
  
А вы пользуетесь торговыми ботами?