Healthy Scepticism

You should naturally be cautious with anyone else's code – the good news is that you can read all of mine to check there's nothing nefarious in there. There's a lot of comments in the code to help it be more readable.
I would DEFINITELY recommend using a separate wallet to your main wallet, certainly while you work out how the bot works and how to set the trade settings.
If you want to be ultra cautious then maybe run it on a virtual machine environment.
I've tried to test the strategies but I can't guarantee they're bulletproof so while you're getting started please run it supervised (i.e. not overnight!)
Once you are confident with what it is doing (and what you are doing) then feel free to get more advanced.

Installation Instructions:

- First of all install nodejs - https://nodejs.org/en/
- Also install git - https://git-scm.com/downloads
- If you're on windows make sure git is added to the path
- Go to a command prompt or terminal and navigate to the folder Documents
  - **`mkdir GitHub`**
  - **`cd GitHub`**
  - **`git clone https://github.com/DrKazza/SimpleBotTrader.git`**
  - **`cd ~\Documents\GitHub\SimpleBotTrader`**
- Now you need to install the Libraries in the same terminal type
  - **`npm install`**
  - *you will see something that says there are some minor vulnerabilities – that's normal and nothing to worry about – these all come from the Pancakeswap library*
- Now that's done you need to set up the initial variables and then start the bot

Setting Up the Bot

Set up Wallet Access

- First of all you need to let the bot have access to your wallet, create a file called **`.env`**
- In this file you need two lines of text:

```
WALLETADDRESS = '0xcABC123ABC123ABC123'
SECRETKEY = 'house cards bakery muppet grizzly head tyre back face'
```

- Just replace the wallet address inside the quotes with your wallet address
- and replace the secretkey words with the words that you got when you created your wallet

Set up Global Variables

- There is a file called `global.paramsPCS.js` in this file there are the parameters that will affect all trades – the only ones that you need to think about really are the "`_keepBNB`" and the "`_gasPrice`" variables – explanations should be in the comments

Set up Trade Specific Variables

- The file called `trading.paramsPCS.js` contains all the details of the trades you are going to set up.
- There are 5 pairs and a template already set up for you – it doesn't matter if you call these '`pair1`' or '`StopTrade`' or '`elephant`' just don't call it '`template`' as that trade will always be ignored
- First of all set the '`activate :`' to **`true`**

- If activate is false then this trade pair will be ignored
- Then define an asset that you'll be buying and one that you'll be selling. You always need to specify the address unless it's BNB in which case just type `'BNB'`
- The two MoonBag variables will just keep back a few tokens that you'll never sell so you'll always have some held back
- Next come the most important variables
  - `tradeType`
  - tradeType specific variables
- Look at the section below on the different trade types to work out which one you want
- Each trade type will have its own specific variables which need to be set – these are described and laid out in the '`template`' trade.
- There's also lots of explanations in the comments.

Starting the Bot

- You're now ready to start the bot – in the same directory as the installation go to your command prompt/terminal and type
  - **`node startBotPCS.js`**
- You'll see a description of each trade that you've requested to trade and if details look right it'll ask you to start – just type Y to start the bot
- The bot will check that you're not trying to put in bad price inputs and also will check that you have the allowance to trade on the PCSv2 Router.
- The Terminal will show the price every 5 seconds when it checks and will start trade if an appropriate price is hit.
- If you want to see a log for what has happened then check in the folder there will be a file called `TradeLogPCS.xxxxxxxx.txt` which has a record of what has gone on for that day. (It doesn't have every price tick but it does have the starting record as well as trade attempts)

Errors

- The bot will generally run quite nicely – if you've entered any inputs incorrectly hopefully you'll get a message explaining what is wrong
- Once in a while an error will be thrown but nothing too frequent

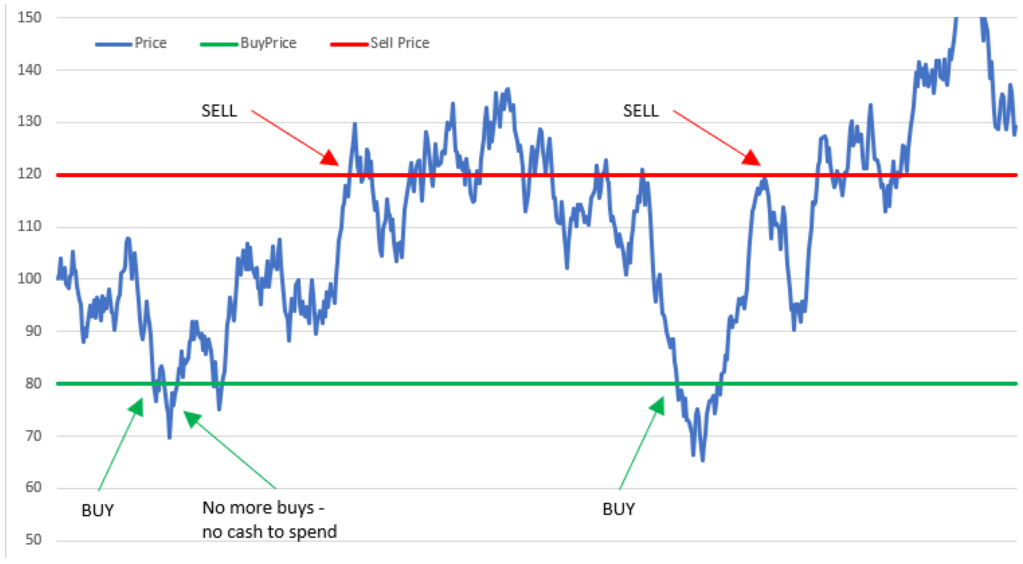
Types of Trade

There are 6 types of trade that the bot can handle and you set them up in the trading.params.js file. You can have different trade types for different assets and as many trades active as you like.
I’ve described them here with some simulated graphs so you can see where the sells and buys are triggered.
Remember that if you’re trading a pair e.g. BUSD/BNB and want to sell with a price of 0.0025 you can just as easily swap the assets to BNB/BUSD and change the price to something more familiar like 400 (USD/BNB) and also swap the sell to a buy.

Also remember that these price limits are the level at which the trade is triggered.
So if you have a sell limit at 110, when the price hits 110 the sale will go thru and may actually end up as a sale at 109.03 for example. These aren’t strict limits.

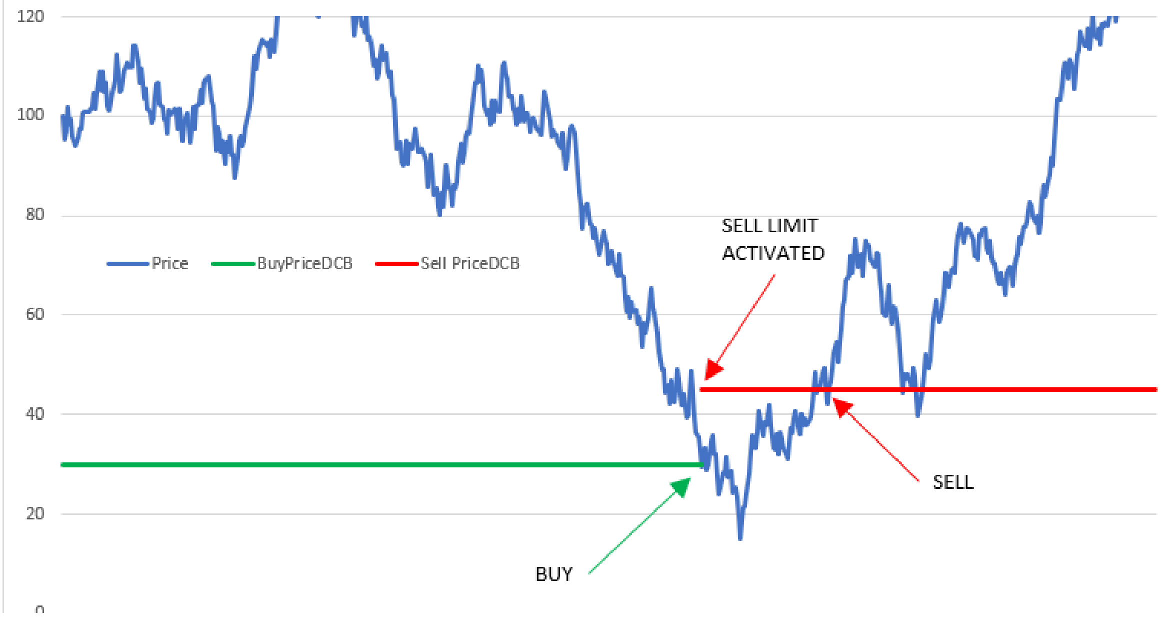
Trade Type 1: `tradeType: 'BUY-SELL-PRICE-LIMITS',`

This is the simplest of all trade types – you specify a price at which you want to buy and a price at which you want to sell. You can use this as a range trade strategy or just if you want to trade at a certain level and then go to bed and leave it to run in the background.



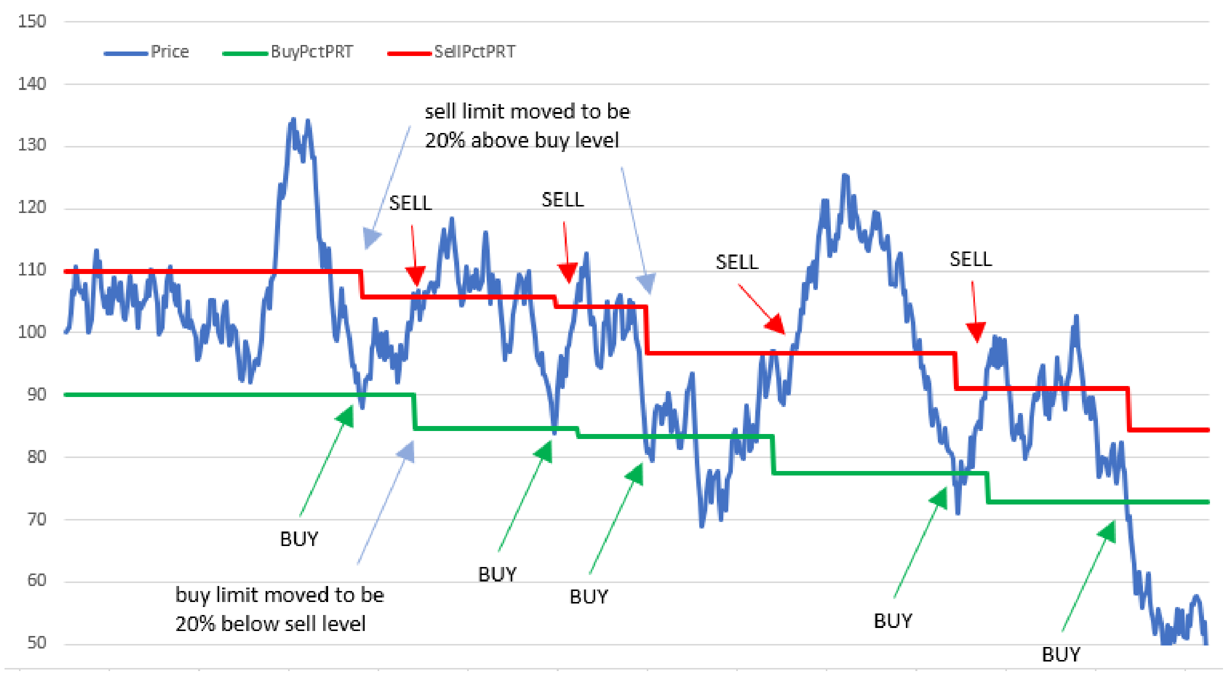
The only variables you need are the **buyPrice** and the **sellPrice**.

Trade Type 2: tradeType: 'DEAD-CAT-BOUNCE',
Very similar to Trade Type 1 you set a level at which you want to buy but the difference is that a sell level isn't set until that buy is reached – so you can actually set a sell below the current market but that will only become active when the market dumps and you buy way below the market.



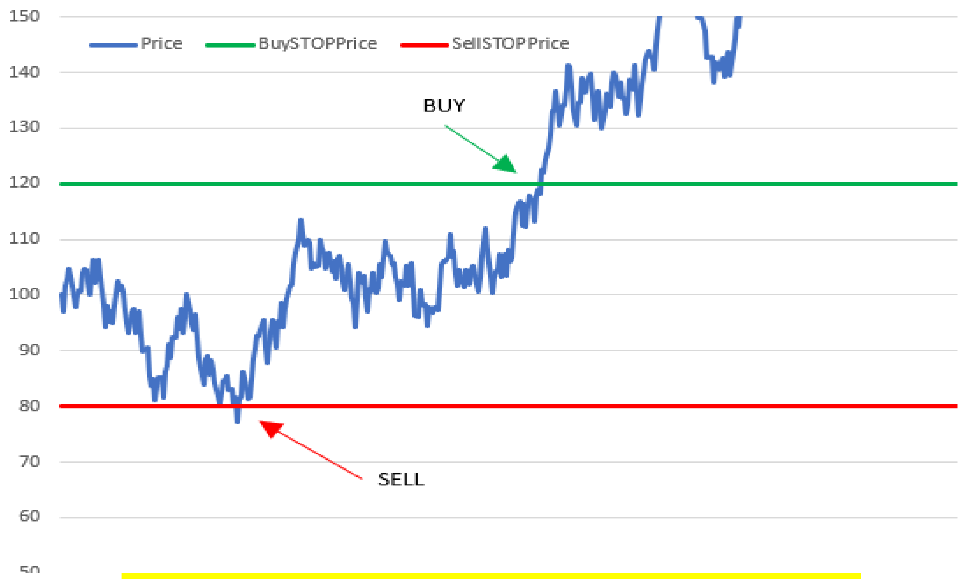
The variables are **buyPriceDCB** which is the level that you buy at when the market drops, the second variable **sellPctDCB** is the percentage increase at which to set the sell level. In the graph above they are 30 and 50 respectively (50 is equal to 50% price increase from 30 – i.e. 45)
This can be a repetitive process but it makes sense to just stop after a single bounce – hence the variable "**stopAfterOneBounceDCB**"

Trade Type 3: tradeType: 'PCT-RANGE-TRADING',
Again similar to Trade Type 1 however this time your ranges move, whenever there is a buy, your sell price is automatically set at a % increase over that purchase (**sellPctPRT**), likewise, when you sell, the next buy is set a % below that sale (**buyPctPRT**)
In this example both the buyPctPRT and sellPctPRT are set to 20%.



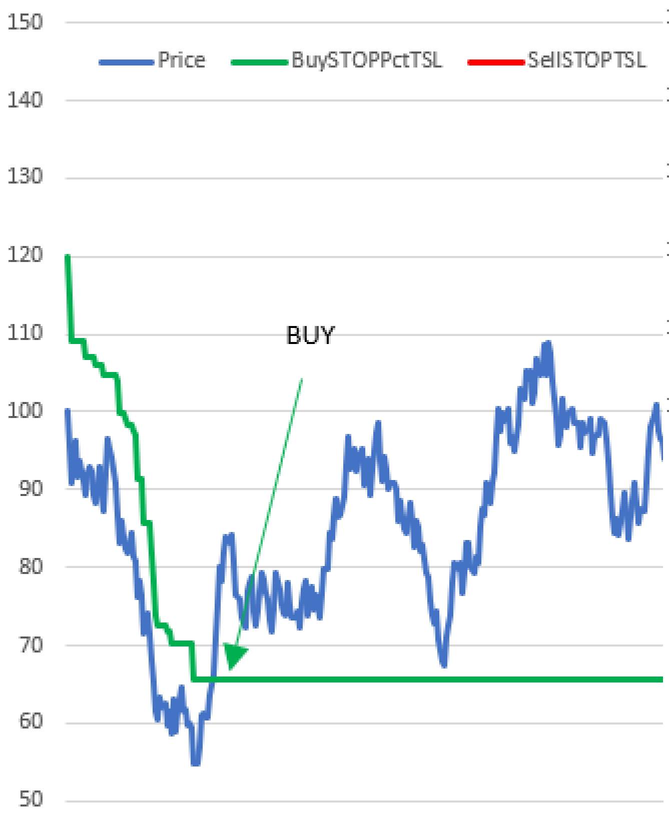
Trade Type 4: tradeType: 'STOP-LOSS',

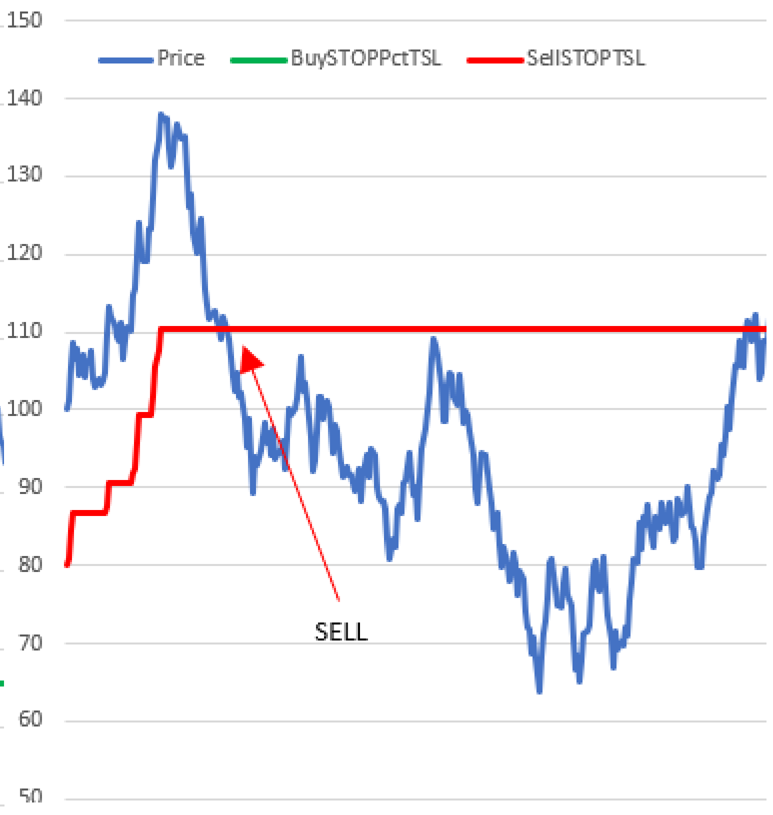
Stop loss trades are incredibly useful to allow you to lock in some profits by selling just as the market collapses, or to buy on the way up as a range is broken. For that reason the **buySTOPPrice** is set above the current price and the **sellSTOPPrice** is set below the market.



Warning – do NOT set a Buy Stop and a Sell Limit on the same pair because there's a chance the Buy Stop will immediately trigger the Sell which will re-trigger the Buy Stop etc etc.
Likewise do not set a Buy Limit and a Sell Stop.
It IS FINE to set a Sell Stop and a Sell Limit (or buy stop and buy limit) because this can be thought of as an "OCO" trade… take profits if this asset goes up, but if it breaks downwards get out quickly.
Also be careful about not setting the Stop Prices too close to each other otherwise a rangebound market could lead to you continually selling low and buying high!

Trade Type 5: tradeType: 'TRAILING-STOP-LOSS',
Trailing Stop loss trades similar to a Stop Loss trade but the level of activation moves with the market if it's moving in your favour. I'd recommend NOT setting both Buy and Sell at the same time.

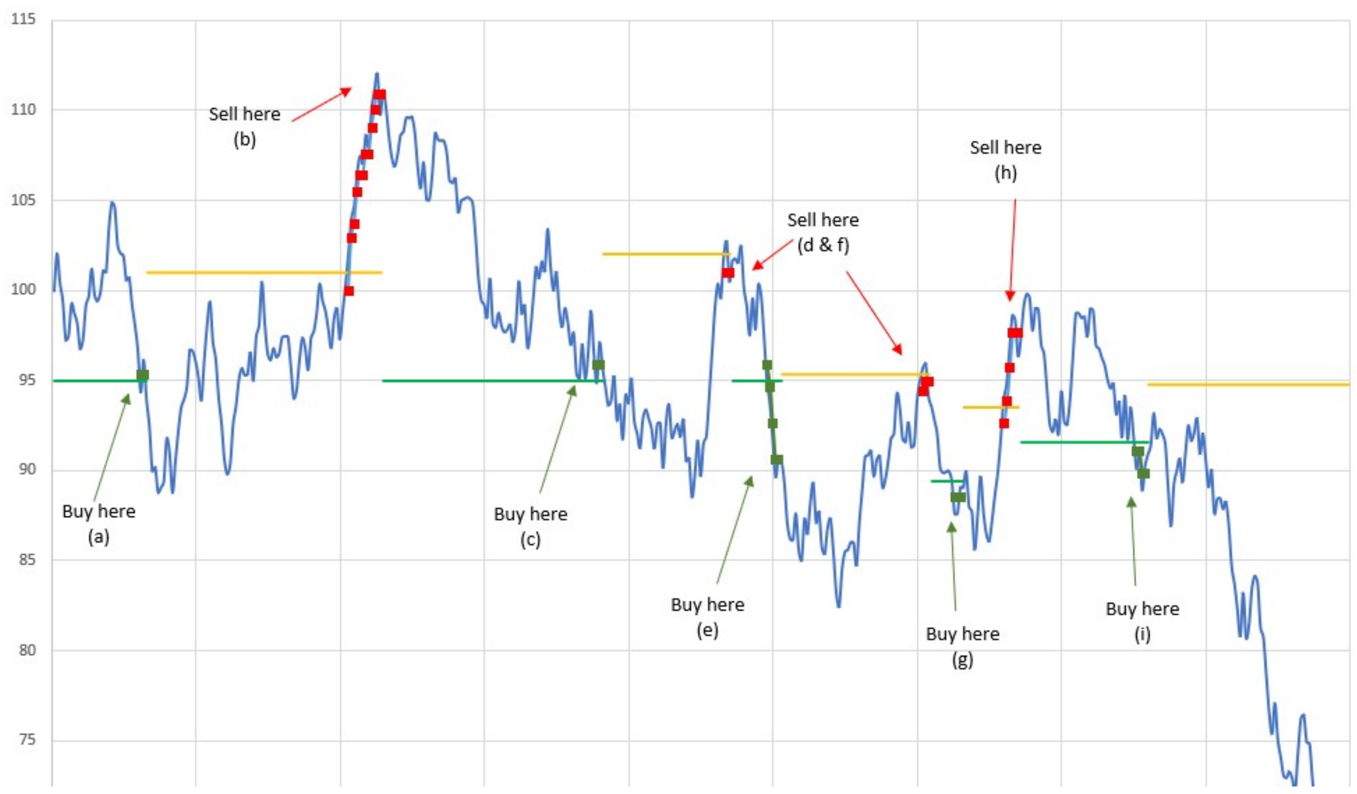




You can see from the above how as the market drops the Buy Stop Price stays 20% above the current market level (**buySTOPPctTSL**), and then buys when the market rallies above that price Likewise on the trailing sell stop your profit taking level increases from 80 up to 110 before getting triggered. Yes you missed the high print but compared to a normal stop you're $30 better off.

Trade Type 6: tradeType: 'SMART-RANGE',
The smart range sets a buy level but will follow the market down and only after it reverses UP will it buy… This is a combination of a trailing stop loss and a range trade.
if you look at the chart below the purchase marked (e) was triggered at 95, it then followed the market down to 90, and then only when the price rose to 91 did it buy (green dots)
A sell is then initiated profitPctSR % above the market and it will sell when you break that level, however it will allow the market to carry on going up until it reverses.
You can see that in sell (d) at 102 a sell was triggered and then it reversed and the sale went through at 101. However on sales (b) and (h) the sale followed the market up selling considerably better.
However you can also see that when you buy in situation (i) the market drops and never recovers so you lose a lot.

You set the initial buy price (in the case below $95), the profit you want to take in % (5 below) and the amount you want to reverse in % before you buy/sell (1 in the example below)


Sell here (b)
Sell here (h)
Sell here (d & f)
Buy here (a)
Buy here (c)
Buy here (e)
Buy here (g)
Buy here (i)