Custom Strategies:

Changes will need to be made to **Bot_Class.py**, **TradingStrats.py** and **app.py** in order to add a custom strategy or new TP/SL function.

TradingStrats.py:

Create a new function to contain your strategy, it should have the following definition at the bare minimum:

def <Strategy Name>(Trade_Direction, current_index, ...(Any indicators you need for the strategy)):

So for example if I wanted to make a simple ema crossover strategy with two EMA's, I would have this definition:

```
def ema_crossover(Trade_Direction, current_index, ema_short, ema_long):
```

inputs: the two required fields Trade_Direction and current_index & then two ema's which we will pass in for our strategy.

Now the Trading logic:

If we wanted to go short when the ema_short crosses below the ema_long, see 1st crossover above.
We would look for a candle where the ema_short is below the ema_long and on the

previous candle the ema_short was above the ema_long. We would enter a short here, by setting Trade_Direction to 0 (indicating a short).

Code:

```
if ema_short[current_index-1] > ema_long[current_index-1] and ema_short[current_index] < ema_long[current_index]:
    Trade_Direction = 0
```

Similarly for the long we would look for a candle where the ema_short is above the ema_long and on the previous candle the ema_short was below the ema_long.

So together this would give the Code:

```
598 def ema_crossover(Trade_Direction, current_index, ema_short, ema_long):
599     if ema_short[current_index-1] > ema_long[current_index-1] and ema_short[current_index] < ema_long[current_index]:
600         Trade_Direction = 0
601     elif ema_short[current_index-1] < ema_long[current_index-1] and ema_short[current_index] > ema_long[current_index]:
602         Trade_Direction = 1
603     return Trade_Direction
604
```

Next in update_indicators() in Bot_Class.Bot we need to add an elif clause that generates the emas for us when self.strategy == 'ema_crossover':

```
elif self.strategy == 'ema_crossover':
    Close = pd.Series(self.Close)
    self.indicators = {"ema_short": {"values": list(ema_indicator(Close, window=20)),
                                     "plotting_axis": 1},
                       "ema_long": {"values": list(ema_indicator(Close, window=50)),
                                    "plotting_axis": 1},
                       }
```

We define each indicator we need in a Dict object like so:

```
{"ema_short":
        {
        "values": list(ema_indicator(Close, window=20)), ← this generates an EMA of window size 20 for us
         "plotting_axis": 1 ← this is needed to graph the trades correctly, see end of document for info on plotting_axis.
        },
…..
}
```

Now in Bot_Class.Make_decision(self): we need to add another elif clause to call our strategy to see if we get a signal to enter a trade:

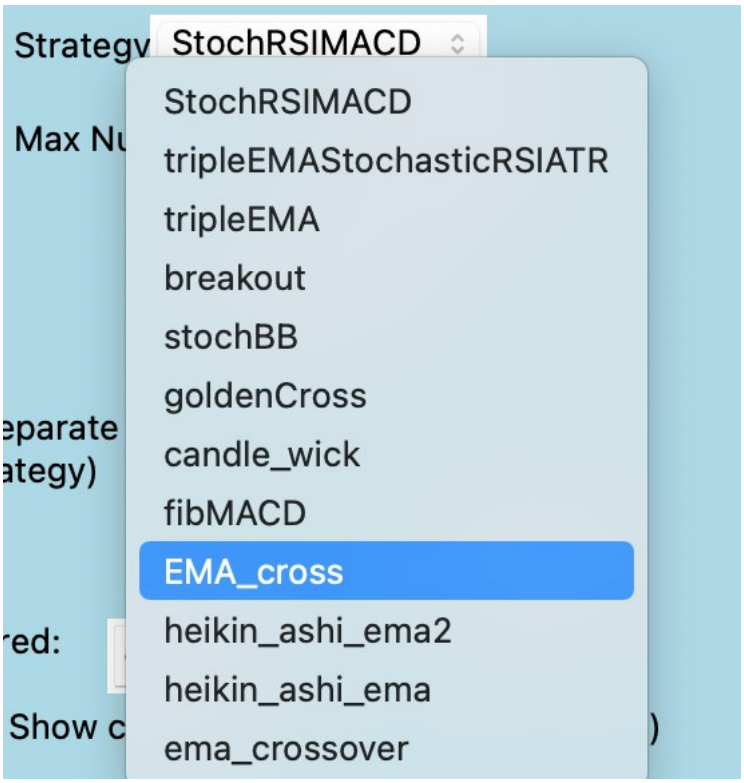
```
elif self.strategy == "ema_crossover":
    Trade_Direction = TS.ema_crossover(Trade_Direction, self.current_index,
                                       self.indicators["ema_short"]["values"],
                                       self.indicators["ema_long"]["values"])
```

You can now run the strategy directly from the backtesting script by replacing the strategy name with “ema_crossover” in __main__.

To get the strategy to display in the UI we need to make a change to app.py:

```
328 strategy_options = ["StochRSIMACD", "tripleEMAStochasticRSIATR", "tripleEMA", "breakout", "stochBB", "goldenCross",
329                     "candle_wick", "fibMACD", "EMA_cross", "heikin_ashi_ema2", "heikin_ashi_ema", "ema_crossover"]
330 strategy = StringVar()
```

It will now display on the dropdown for the UI:



← see here

plotting_axis:

This is needed for the new graphing functionality I've added, theres not much to this other than selecting the correct axis for the trade graph to display nicely:
Axis 1: this is where the candle sticks are displayed, so if you have any indicators you want to display overlapping the candles you can that indicator a "plotting_axis": 1.
Axis 2: this is where volume is displayed, I just have it in as a line graph but it isn't hard to change to bars if thats something you wanted to do. If you want any indicators to overlap with the volume chart you would give them a "plotting_axis": 2.
Any Axis higher than 1 is found below the main candlestick chart:
So if you want overlap between any indicators say a fast and slow stochastic you might give them both "plotting_axis": 3 so that they are separated from the candles but still overlap with each other.

See Image on plotting_axis:

plotting_axis: 1
plotting_axis: 2
plotting_axis: 3
plotting_axis: 4
plotting_axis: 5