Custom Strategies & TP/SL functions:

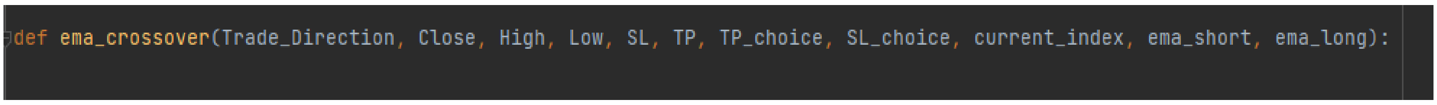
Changes will need to be made to **Bot_Class.py**, **TradingStrats.py** and **app.py** in order to add a custom strategy or new TP/SL function.

TradingStrats.py:

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

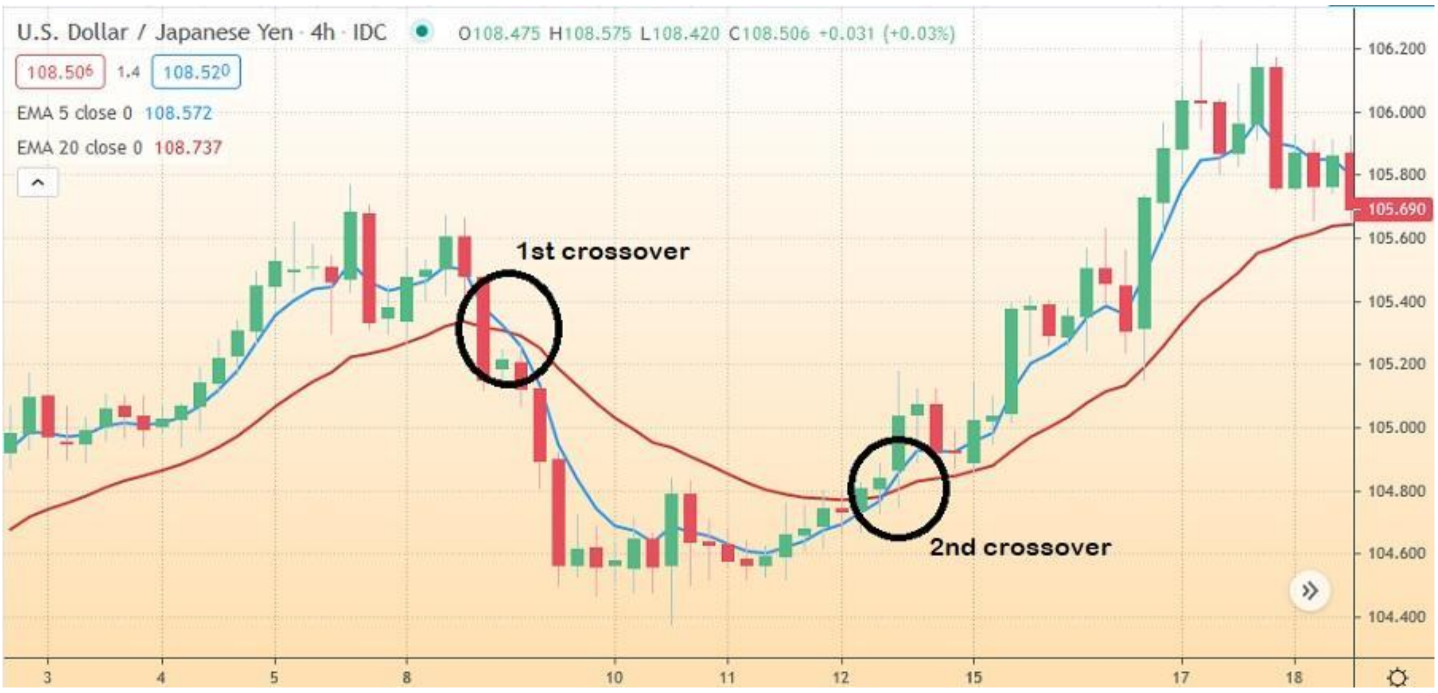
def <Strategy Name>(Trade_Direction, Close, High, Low, SL, TP, TP_choice, SL_choice, index, …(Any indicators you need for the strategy)):

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



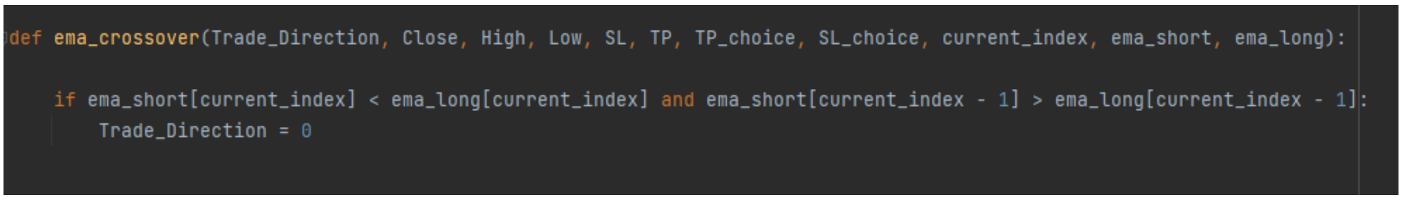
```
def ema_crossover(Trade_Direction, Close, High, Low, SL, TP, TP_choice, SL_choice, current_index, ema_short, ema_long):
```

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).

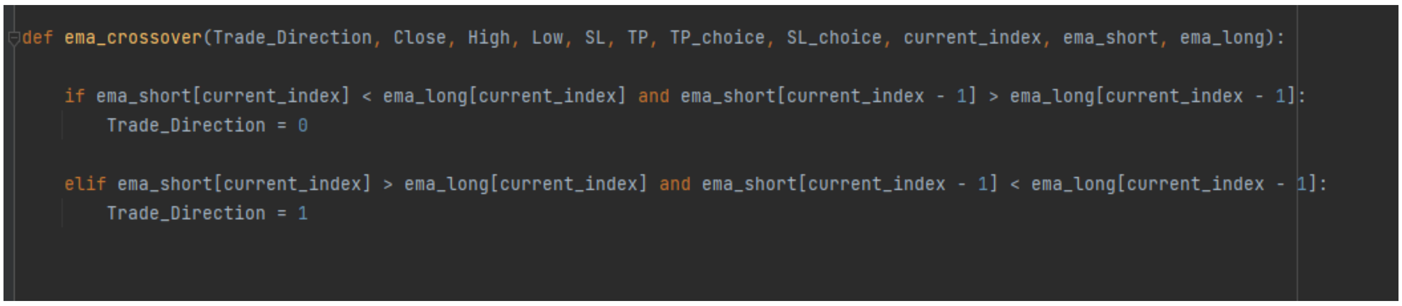
Code:



```python
def ema_crossover(Trade_Direction, Close, High, Low, SL, TP, TP_choice, SL_choice, current_index, ema_short, ema_long):

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

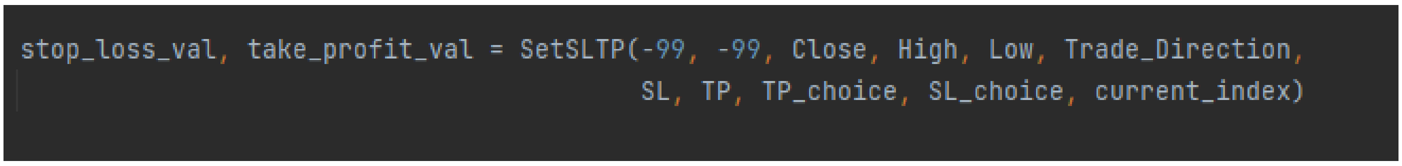
Similarly for the long we would look for a candle where the ema_short is above the ema_long and on the previous candle the ema_short was below the ema_long.

So together this would give the Code:



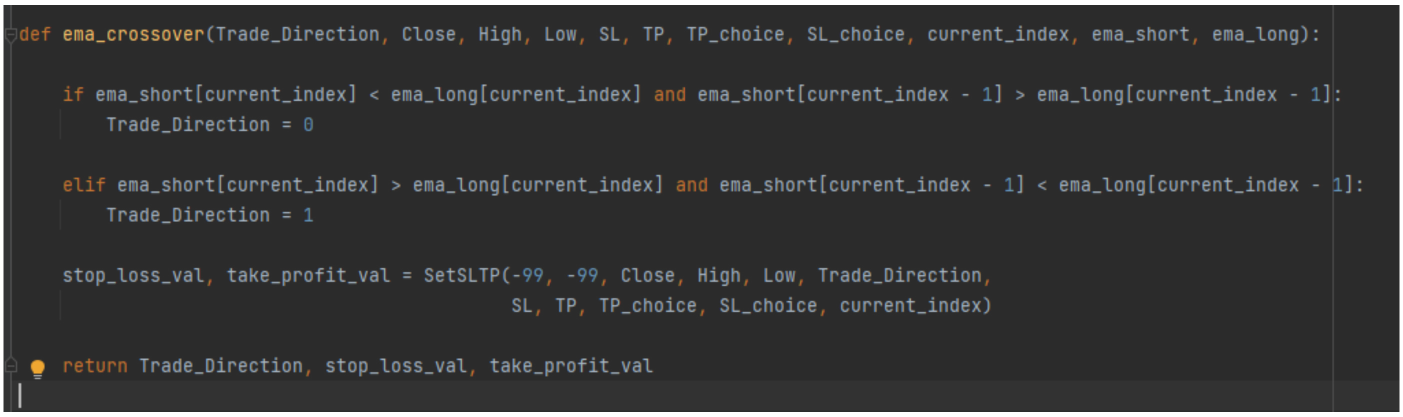
```python
def ema_crossover(Trade_Direction, Close, High, Low, SL, TP, TP_choice, SL_choice, current_index, ema_short, ema_long):

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

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

Now setting up the TP and SL, simply add the following line of code to work with the TP and SL functions that come out of the box:



```python
stop_loss_val, take_profit_val = SetSLTP(-99, -99, Close, High, Low, Trade_Direction,
                                         SL, TP, TP_choice, SL_choice, current_index)
```

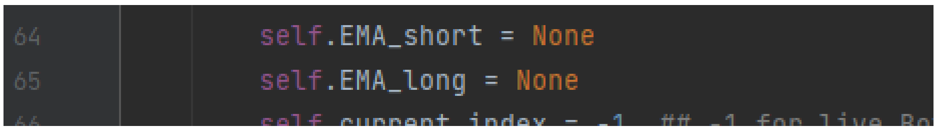
The final product, we need to return the Trade_Direction, stop_loss_val and take_profit_val:



```python
def ema_crossover(Trade_Direction, Close, High, Low, SL, TP, TP_choice, SL_choice, current_index, ema_short, ema_long):

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

    elif ema_short[current_index] > ema_long[current_index] and ema_short[current_index - 1] < ema_long[current_index - 1]:
        Trade_Direction = 1

    stop_loss_val, take_profit_val = SetSLTP(-99, -99, Close, High, Low, Trade_Direction,
                                             SL, TP, TP_choice, SL_choice, current_index)

    return Trade_Direction, stop_loss_val, take_profit_val
```

Next add this to Bot_Class.py:

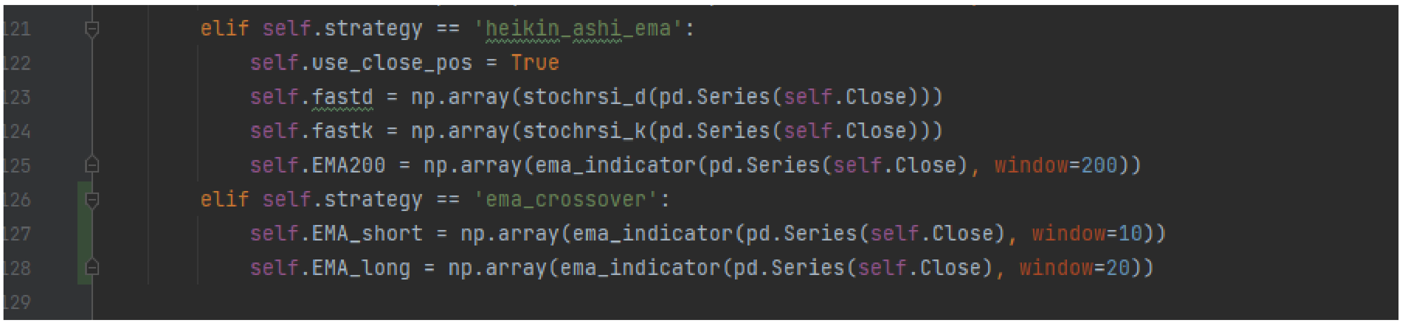
Firstly in the __init__(….) function we need to define our emas.



```python
        self.EMA_short = None
        self.EMA_long = None
        self.current_index = -1  ## -1 for live Bo
```

The default bot already has two variables for ema_short and ema_long so we can just reuse them.

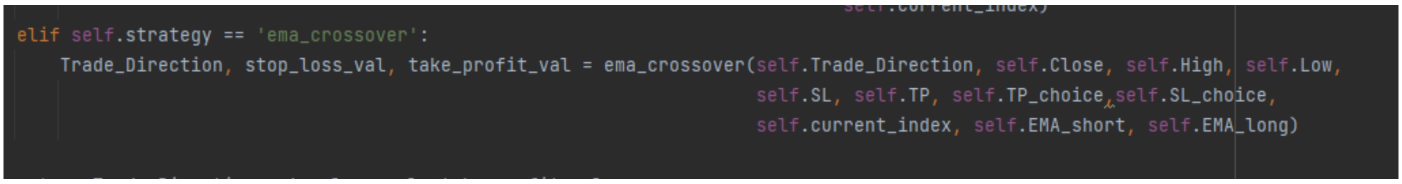
Next in update_indicators() we need to add an elif clause that generates the emas for us when self.strategy == 'ema_crossover':



```python
        elif self.strategy == 'heikin_ashi_ema':
            self.use_close_pos = True
            self.fastd = np.array(stochrsi_d(pd.Series(self.Close)))
            self.fastk = np.array(stochrsi_k(pd.Series(self.Close)))
            self.EMA200 = np.array(ema_indicator(pd.Series(self.Close), window=200))
        elif self.strategy == 'ema_crossover':
            self.EMA_short = np.array(ema_indicator(pd.Series(self.Close), window=10))
            self.EMA_long = np.array(ema_indicator(pd.Series(self.Close), window=20))
```

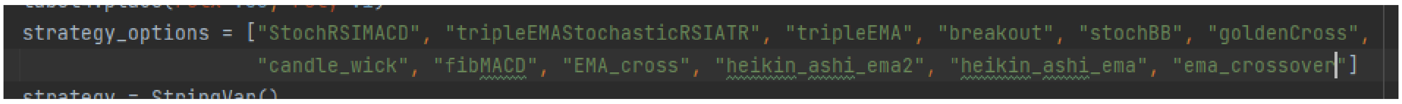
update_indicators() gets called when a new candle comes in for the live bot and once at the start of a backtest to calculate the indicators.

Similarly we add an elif clause in Make_Decision() now feeding the relevant variables to our strategy:



```python
elif self.strategy == 'ema_crossover':
    Trade_Direction, stop_loss_val, take_profit_val = ema_crossover(self.Trade_Direction, self.Close, self.High, self.Low,
                                                                    self.SL, self.TP, self.TP_choice, self.SL_choice,
                                                                    self.current_index, self.EMA_short, self.EMA_long)
```

Finally, in app.py we need to add ema_crossover to the strategy_options array:



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

And that's all we need to do to add a new strategy.