# My Hummingbot Trading Strategy: mystategy.py

## Overview

Hey! This project is all about building a trading bot using Hummingbot, an awesome open-source framework for creating high-frequency trading bots. I built a market-making strategy through a custom script named mystategy.py. The strategy is focused on the ETH-USDT trading pair, where it places both buy and sell orders around the mid-market price to earn profit from the spread.  
  
- Order Size: 0.01 ETH  
- Spread: 0.001%  
- Refresh Interval: Every 15 seconds  
- Price Source: Mid-price  
  
The goal is to maintain liquidity and generate consistent returns in both paper trading and, eventually, live market conditions.

## Technical Details

### Setup

I used a config file named conf\_simple\_pmm\_1.yml, which was created using Hummingbot's create --script-config command. Key parameters:  
  
- Trading Pair: ETH-USDT  
- Order Size: 0.01 ETH  
- Bid/Ask Spread: 0.001%  
- Order Refresh Time: 15 seconds  
- Price Source: Mid-price

### Coding It

My custom script mystategy.py extends Hummingbot's ScriptStrategyBase class. I used methods like place\_market\_order to execute trades. I referenced the official Hummingbot GitHub repository:  
https://github.com/hummingbot/hummingbot

### Issues Faced & Fixes

- Error 1: "Non-hexadecimal digit found"  
 Fix: Identified and corrected invalid input in the YAML config file.  
  
- Error 2: "No module named 'hummingbot.core.data\_type.order\_type'"  
 Fix: Reinstalled Hummingbot using ./uninstall and ./install after pulling the latest changes via git pull.  
  
- Other Setup: Exchange connection via API keys was configured, and a stable internet connection was maintained to ensure connectivity.

## Results

- Mode: Paper Trading  
- Funds Used: 0.01 ETH (simulated)  
- Trades Executed: 0 (initial run)  
- P&L: 0.00 USDT  
  
Although no trades were executed during the short test (3–10 minutes), the bot ran smoothly after resolving the issues, and it successfully connected to the exchange and initiated order processes.

## Future Improvements

- Implement robust error handling and recovery mechanisms.  
- Adjust spread and order size dynamically based on market volatility.  
- Deploy the strategy with live funds (small amount initially) to assess real-world performance.

## Conclusion

Working on this Hummingbot project was a fun and rewarding experience. It helped me level up my skills in:  
  
- Using GitHub repositories  
- Working with terminal commands and package setups  
- Debugging and problem-solving in real-time  
  
I now have a solid grasp of building and managing market-making bots, which makes me a strong fit for roles in DeFi and algorithmic trading environments.