

- Market Regime Detec...
- Advanced Volatility R...
- Market Correlation Re...
- Christian Kolomagic's ...
- Pattern Analysis :AD...
- Volatility Analysis
- ... 간략히 보기 ^
- GPT 탐색

오늘  
Fractal Trail Indicator Explai...

어제  
환불 요청 및 안내

- 지난 7일
- Volatility Analysis and Strategy
- Movedo Favicon 설정 방법
- DMI Volatility Analysis Predictio
- 이스라엘 민족과 기도문

```
// This Pine Script™ code is subject to the terms of the
// © UAlgo
//@version=5
indicator(title='Fractal Trail [UAlgo]', shorttitle='Frac

bool showWilliamsFractals = input.bool(defval=true, title
bool showWilliamsStops = input.bool(defval=true, title='
int williamsLeftRange = input.int(defval=2, minval=1, ma
int williamsRightRange = input.int(defval=2, minval=1, m
float williamsStopBuffer = input.float(defval=0, minval=0
string flipInput = input.string(title='Trail Invalidator
color upTrailColor = input.color(defval=color.rgb(40, 218
color dnTrailColor = input.color(defval=color.new(color.l
color fillColorUp = color.new(color.rgb(40, 218, 150), 80
color fillColorDn = color.new(color.blue, 80)

f_IsWilliamsFractal(_leftRange, _rightRange, _type) =>
    _isFractal = _type == 'high' and high[_rightRange] >
    _fractalValue = _isFractal and _type == 'high' ? high
    [_isFractal, _fractalValue]

[isHighFractal, highFractalPrice] = f_IsWilliamsFractal(v
[isLowFractal, lowFractalPrice] = f_IsWilliamsFractal(wil

isHighFractal := isHighFractal[1] ? false : isHighFractal
isLowFractal := isLowFractal[1] ? false : isLowFractal

highFractalOffset = 0 - williamsRightRange
color white10 = color.new(color.white, 10)
plotshape(isHighFractal and showWilliamsFractals, title='
lowFractalOffset = 0 - williamsRightRange
plotshape(isLowFractal and showWilliamsFractals, title='

f_addPercentBuffer(_input, _buffer, _direction) =>
    _direction == 'plus' ? _input * (1 + _buffer / 100) :

lowFractalPriceBuffered = f_addPercentBuffer(lowFractalPr
highFractalPriceBuffered = f_addPercentBuffer(highFractal

f_persistAndReset(_trigger, _source) =>
    var float _output = 0.0
    _output := _trigger ? _source : _output[1]
    _output

longStopPrice = f_persistAndReset(isLowFractal, lowFract
shortStopPrice = f_persistAndReset(isHighFractal, highFr

f_trail(_source, _trail, _direction) =>
    _direction == 'down' and _source >= _trail[1] ? _trai

var float longStopPriceTrail = longStopPrice
var float shortStopPriceTrail = shortStopPrice

shortStopPriceTrail := f_trail(shortStopPrice, shortStopP
longStopPriceTrail := f_trail(longStopPrice, longStopPric

f_flip(_flipInput, _longTrail, _shortTrail, _longReset, _
    var bool _flipLongNow = false
    var bool _flipShortNow = false
    var bool _isLong = true
    var bool _isShort = true
    float _flipLongSource = _flipInput == 'Close' ? close
    float _flipShortSource = _flipInput == 'Close' ? clo
    _flipLongNow := _isShort[1] and _flipLongSource > _st
    _flipShortNow := _isLong[1] and _flipShortSource < _
    _flipLongNow := _flipShortNow and _flipLongNow and cl
    _flipShortNow := _flipLongNow and _flipShortNow and c
    _isLong := _flipLongNow ? true : _flipShortNow ? fals
    _isShort := _flipShortNow ? true : _flipLongNow ? fal
    _longTrailOutput = _longTrail
    _shortTrailOutput = _shortTrail
    _longTrailOutput := _isLong and not _isLong[1] ? _lor
    _shortTrailOutput := _isShort and not _isShort[1] ? _
    float _longTrailPlot = _isLong ? _longTrailOutput : _
    float _shortTrailPlot = _isShort ? _shortTrailOutput
    [_longTrailOutput, _shortTrailOutput, _longTrailPlot,

f_getFlipResetWilliamsLong(_longTrail, _buffer, _isLowFr
    _barIndexWhenLastFractalConfirmed = ta.valuewhen(_isL
    _barsSinceLastFractalConfirmed = bar_index - _barInde
    int _barsToGoBack = _barsSinceLastFractalConfirmed +
    float _lowestLow = low
    for i = 0 to _barsToGoBack by 1
        _lowestLow := math.min(low[i], _lowestLow)
        _lowestLow
    _lowestLowAdjusted = _lowestLow * (1 - _buffer / 100);
    _lowestLowAdjusted
```

```

f_getFlipResetWilliamsShort(_shortTrail, _buffer, _isHigh
_barIndexWhenLastFractalConfirmed = ta.valuewhen(_is
_barSinceLastFractalConfirmed = bar_index - _barInde
int _barsToGoBack = _barsSinceLastFractalConfirmed +
float _highestHigh = high
for i = 0 to _barsToGoBack by 1
    _highestHigh := math.max(high[i], _highestHigh)
    _highestHigh
    _highestHighAdjusted = _highestHigh * (1 + _buffer /
    _highestHighAdjusted

longStopPrice := f_getFlipResetWilliamsLong(longStopPrice
shortStopPrice := f_getFlipResetWilliamsShort(shortStopPr

[longStopPriceTrailTemp, shortStopPriceTrailTemp, longSt

longStopPriceTrail := longStopPriceTrailTemp
shortStopPriceTrail := shortStopPriceTrailTemp

shortStopPriceTrailPlotDisplay = showWilliamsStops ? shor
longStopPriceTrailPlotDisplay = showWilliamsStops ? longS

plot(shortStopPriceTrailPlotDisplay, 'Williams Trailing S
plot(shortStopPriceTrailPlotDisplay, 'Williams Trailing S
plot(longStopPriceTrailPlotDisplay, 'Williams Trailing St
plot(longStopPriceTrailPlotDisplay, 'Williams Trailing St

pricefill = plot(hl2, 'Midline', color=color.gray, displ

isUptrend = close < shortStopPriceTrailPlotDisplay
isDowntrend = close > longStopPriceTrailPlotDisplay

fillColor = isUptrend ? color.new(fillColorDn, 85) : col
fillPrice = isUptrend ? shortStopPriceTrailPlotDisplay :


fill(plot(fillPrice, title='Williams Trailing Stop Fill F

alertCrossLongStop = na(longStopPriceTrailPlot) and not r
alertCrossShortStop = na(shortStopPriceTrailPlot) and not
alertcondition(alertCrossLongStop, title='Crossed Willia
alertcondition(alertCrossShortStop, title='Crossed Willia

alertcondition(isHighFractal, title='High Printed', mess
alertcondition(isLowFractal, title='Low Printed', message

study this code

```

🔗 This Pine Script is a sophisticated implementation that identifies and plots Williams Fractals on a price chart and also implements a dynamic trailing stop mechanism. Let me break down the main components and logic used here: 

### 1. Indicator Setup

The indicator is titled "Fractal Trail [UAlgo]". It includes the following functionalities:

- **Williams Fractals Visualization:** Marks each identified fractal with a triangle on the chart.
- **Williams Trail:** Provides a dynamic trailing stop, based on the latest identified fractal level.

The following parameters can be customized by the user:

- **Fractal Range:** You can adjust how many bars to the left and right are checked to identify a fractal (i.e., peak or trough).
- **Trail Buffer:** Sets a margin of safety (percentage-based) between the fractal price and the trailing stop.

### 2. Fractal Calculation

The core of this script revolves around fractal detection using Williams Fractals. A fractal is an indicator of a potential trend reversal point:

- **Fractal Detection Logic:**
  - A high fractal is identified if the current bar's high is the highest of the bars in the specified range.
  - A low fractal is identified if the current bar's low is the lowest of the bars in the specified range.

The function `f_IsWilliamsFractal()` is used to determine if a high or low fractal exists at a particular bar. Fractals are plotted using `triangles` to indicate potential reversal points.

### 3. Adding Trail Buffer

The trailing stop is calculated by adding (or subtracting) a user-defined buffer percentage from the fractal price. This serves as a safety margin. The function `f_addPercentBuffer()` calculates this buffer.

### 4. Persist and Reset Mechanism

The function `f_persistAndReset()` ensures that the trailing stop price persists until a new fractal is confirmed at which point it resets to the new level.

is confirmed, at which point it resets to the next level.

## 5. Trail Calculation


The trailing stops are calculated for both long and short positions:

- **Long Stop Price:** Calculated based on low fractals.
- **Short Stop Price:** Calculated based on high fractals.

The script maintains the trailing stops using a function `f_trail()`. Depending on the price action, it updates the trailing stops to follow the movement of price, helping to maintain a logical level for stop loss placement.

## 6. Flip Logic

The function `f_flip()` is used to determine when the trend should switch from long to short, or vice versa, based on the current price crossing over the stop levels:

- **Flip Input:** Either uses closing prices or  wick (high/low) of the candle to determine if a switch should happen.
- **Trail Output:** Once a trend switch is identified, it resets the appropriate trailing stop to the correct level.

## 7. Fill Colors and Alerts

The script fills the area between the price and the trailing stop using specific colors to indicate whether the trend is in an uptrend or downtrend:


- **Fill Color:** Green (uptrend) or blue (downtrend).
- **Alerts:** Alerts are triggered when:
  - The price crosses the trailing stop level.
  - A new high or low fractal is confirmed.

## 8. Fractal Trail Visualization

The script visually plots:

- **Triangles** above or below bars to mark fractal points.
- **Lines** to represent trailing stop prices for both long and short positions.
- **Color Fill** to indicate the trend direction between trailing stops and the price.

## Key Functions Overview

1. `f_IsWilliamsFractal()`: Determines if the current bar is a high or low fractal.
2. `f_addPercentBuffer()`: Adds a buffer to  fractal price.
3. `f_persistAndReset()`: Persists the stop level until a new fractal is formed.
4. `f_trail()`: Maintains the trailing stop at the appropriate level.
5. `f_flip()`: Handles the transition between trends.
6. `f_getFlipResetWilliamsLong()` and `f_getFlipResetWilliamsShort()`: Calculate the reset levels for the trailing stops when a trend switch occurs.

## Summary

This script implements a technical analysis tool that identifies key reversal points (fractal levels) on the chart and then creates a dynamic trailing stop based on these levels. The trailing stop is continuously adjusted and plotted to show where the current stop level would be, providing traders with a visual representation of possible stop levels that adapt dynamically with price movements.

The use of different visual elements (triangles, lines, and fills) helps make it easy to identify:

- Where fractals have formed.
- The current stop levels for both long and short positions.
- The prevailing trend direction.

This tool can be helpful for traders who want to follow a systematic approach to trailing stops, adapting to new price action without manually adjusting stops all the time.

If you need a more specific part of the code explained in detail or a comparison to a Python implementation, feel free to ask!



Fractal ... 쉽게 설명해줘



프랙탈(fractal)은 차트에서 **가격이 방향을 바꿀 가능성이 있는 중요한 지점**을 나타내는 신호예요. 쉽게 말해, **위나 아래로 크게 움직일 가능성이 있는 가격의 꼭짓점**을 찾는 거라고 생각하면 돼요.

예를 들어, 가격이 계속 오르다가 한 지점에서 더 이상 오르지 않고 내려가기 시작하면, 그 지점을 **상위 프랙탈**이라고 해요. 반대로, 가격이 계속 떨어지다가 멈추고 다시 오르기 시작하는 지점은 **하위 프랙탈**이라고 해요.

프랙탈은 보통 **이전 몇 개의 캔들**을 기준으로 만들어져요. 예를 들어, 현재 봉이 양쪽으로 2개씩, 총 5개의 봉 중에 제일 높거나 낮을 때 프랙탈이 생기는 거예요.

Trail Buffer: Sets a margin of safety (percentage-based) between the

Trail Buffer(트레일 버퍼)는 **프랙탈 가격과 실제로 설정하는 손절선(트레일링 스탱) 사이에 안전한 거리를 두기 위한 여유 공간**이라고 생각하면 돼요.

예를 들어, 프랙탈이 100원에서 형성됐다고 가정할 때, 트레일 버퍼를 5%로 설정했다면 손절선은 **100원의 5%만큼 떨어진 95원**이 되는 거예요. 이렇게 하면 **가격 변동에 조금 더 안전하게 대처**할 수 있어요. 가격이 잠깐 흔들리더라도 바로 손절되지 않고 조금 더 여유 있게 지켜보는 거죠.



Yes I need a comparison to a Python implementation. Make me a wisest analyzer in the world , will you?

메모리 업데이트됨

Absolutely! Let's break down how we can implement a similar approach in Python compared to the Pine Script you provided. I'll focus on how each part of the Pine Script can be translated into Python, making you a step closer to becoming the wisest analyzer in the world.

Here's how we can compare the different functions:

### 1. Setup & Libraries in Python

In Pine Script, you don't need to explicitly import libraries, but in Python, we need to import relevant libraries such as Pandas for data manipulation and NumPy for numerical operations.

```
python 코드 복사
import pandas as pd
import numpy as np
```

### 2. Detecting Fractals

In Pine Script, `f_IsWilliamsFractal()` is used to determine if a certain point is a high or low fractal. Here's how you can implement a similar logic in Python.

```
python 코드 복사
def find_fractals(df, left_range=2, right_range=2):
    high_fractals = []
    low_fractals = []

    for i in range(left_range, len(df) - right_range):
        is_high_fractal = all(df['High'][i] > df['High'][i - j] for j in range(1, left_range))
        is_low_fractal = all(df['Low'][i] < df['Low'][i - j] for j in range(1, right_range))

        high_fractals.append(df['High'][i] if is_high_fractal else np.nan)
        low_fractals.append(df['Low'][i] if is_low_fractal else np.nan)

    return pd.Series(high_fractals, index=df.index[left_range:len(df) - right_range]), pd.
```

#### • Explanation:

- In this function, `df` is a DataFrame that contains the high and low prices.
- We iterate through the price data and check if a specific price point is higher (or lower) than the surrounding bars based on `left_range` and `right_range` to determine if it's a fractal.
- The result is returned as two separate Pandas series for high and low fractals.

### 3. Adding the Trail Buffer

The buffer in Pine Script is a safety margin between the fractal price and the trailing stop. Here's a similar implementation in Python:

```
python 코드 복사
def add_buffer(fractal_prices, buffer_percent, direction='minus'):
    if direction == 'plus':
        return fractal_prices * (1 + buffer_percent / 100)
    elif direction == 'minus':
        return fractal_prices * (1 - buffer_percent / 100)
    else:
        return fractal_prices
```

#### • Explanation:

- This function takes the fractal prices and adjusts them by the `buffer_percent`.
- It either increases (for high trailing stop) or decreases (for low trailing stop) the fractal prices to create a margin of safety.

### 4. Trailing Stop Calculation

The trailing stop is updated as new fractals are found. This corresponds to the `f_trail()` function in Pine Script.

```
python 코드 복사
def update_trailing_stop(fractal_prices, previous_trail_price_data, direction='up'):
```

```

trail = []
for i in range(len(fractal_prices)):
    if direction == 'down' and price_data[i] >= previous_trail:
        trail.append(previous_trail)
    elif direction == 'up' and price_data[i] <= previous_trail:
        trail.append(previous_trail)
    else:
        trail.append(fractal_prices[i])
    previous_trail = trail[-1]
return pd.Series(trail, index=fractal_prices.index)

```

• **Explanation:**

- This function calculates the trailing stop level for each price point.
- It updates based on whether the price crosses above or below the previously calculated trailing stop.

### 5. Trend Flip Logic

The `f_flip()` function is the core logic for determining when the trend direction changes based on whether price crosses trailing stops.

```

python
def flip_trend(price_data, long_trail, short_trail):
    is_long = True
    long_trail_output = []
    short_trail_output = []

    for i in range(len(price_data)):
        if is_long and price_data[i] < short_trail[i]:
            is_long = False
        elif not is_long and price_data[i] > long_trail[i]:
            is_long = True

        long_trail_output.append(long_trail[i] if is_long else np.nan)
        short_trail_output.append(short_trail[i] if not is_long else np.nan)

    return pd.Series(long_trail_output, index=price_data.index), pd.Series(short_trail_output, index=price_data.index)

```

• **Explanation:**

- This function tracks whether the trend is currently long or short.
- It outputs the appropriate trailing stop level based on whether the trend is currently long or short.

### 6. Plotting in Python

Unlike Pine Script, Python doesn't have a direct chart interface with trading platforms. However, you can visualize your trailing stops using libraries like Matplotlib.

```

python
import matplotlib.pyplot as plt

def plot_trailing_stops(df, long_trail, short_trail):
    plt.figure(figsize=(14, 7))
    plt.plot(df.index, df['Close'], label='Price', color='black')
    plt.plot(long_trail.index, long_trail, label='Long Stop Price', linestyle='--', color='red')
    plt.plot(short_trail.index, short_trail, label='Short Stop Price', linestyle='--', color='green')
    plt.legend()
    plt.show()

# Assuming 'df' is your DataFrame containing the price data.
plot_trailing_stops(df, long_stop_price_trail, short_stop_price_trail)

```

• **Explanation:**

- Here, we use Matplotlib to plot the original price data along with the trailing stops.
- This gives a similar visualization to what is displayed in Pine Script on TradingView.

### Summary

- In **Pine Script**, the fractals are identified, stops are adjusted, and conditions are evaluated all in one script.
- In **Python**, we create separate functions to calculate fractals, add buffer, calculate trailing stops, flip trends, and plot data. This modular approach helps in better customization and analysis.

By understanding and implementing each component of this Pine Script in Python, you'll become adept at creating complex technical analysis models programmatically. It also allows you to backtest and fine-tune trading strategies with the immense power of Python libraries, giving you an edge in the world of quantitative analysis.

Is there a particular part you'd like to dive deeper into? Or perhaps you'd like to focus on building a complete backtesting solution for these types of indicators?

이 코드들을 전체로 통합하면 어떤 코드가 될까?

물론이죠! 지금까지 설명드린 개별 함수들을 모두 통합하면, Python으로 트레이딩 분석을 위해 전제적인 절차를 진행하는 하나의 코드가 됩니다. 이 코드는 데이터 불러오기, 프랙탈 탐지, 트레이일 버퍼 추가, 트레이일링 스탱 계산, 트렌드 플립, 그리고 차트 시각화까지 한 번에 진행할 수 있도록 구현됩니다. 이를 통해 Pine Script의 기능을 Python으로 완전히 구현해 보겠습니다.

Python Trailing Stop Script

위 코드는 각 단계별로 설명드린 내용을 통합하여 Python에서 Williams 프랙탈을 탐지하고, 트레이일 버퍼를 설정하며, 트레이일링 스탱을 계산하고, 트렌드 플립 로직을 적용하는 과정을 모두 포함하고 있어요. 이제 이를 사용해 차트를 시각화하고 트렌드를 분석할 수 있습니다.

다음 단계로, 이 코드를 실시간 데이터에 적용하거나, 더 나아가 백테스팅 기능을 추가해 볼 수 있습니다. 필요하시면 더 도와드릴 테니 언제든지 말씀해 주세요!

```
from flask import Flask, request, jsonify, render_template
import pandas as pd
import numpy as np
import yfinance as yf
from datetime import datetime, timedelta
import matplotlib
matplotlib.use('Agg') # Use a non-GUI backend for Matplotlib
import matplotlib.pyplot as plt
import io
import base64
import os

app = Flask(__name__)

# Function to load data from Yahoo Finance
def load_stock_data(ticker, start_date, end_date):
    stock = yf.download(ticker, start=start_date, end=end_date)
    stock.reset_index(inplace=True)
    return stock

# Function to format the ticker for Yahoo Finance
def format_ticker(ticker):
    if ticker.isdigit(): # Korean stock ticker
        return ticker + ".KS"
    return ticker # US or other tickers

# DMI and ADX Calculation
def calculate_dmi_adx(data, period=14):
    high = data['High']
    low = data['Low']
    close = data['close']

    plus_dm = high.diff()
    minus_dm = low.diff()

    plus_dm[plus_dm < 0] = 0
    minus_dm[minus_dm > 0] = 0

    tr1 = high - low
    tr2 = abs(high - close.shift(1))
    tr3 = abs(low - close.shift(1))
    tr = pd.concat([tr1, tr2, tr3], axis=1).max(axis=1)

    tr_smooth = tr.rolling(window=period).sum()
    plus_dm_smooth = plus_dm.rolling(window=period).sum()
    minus_dm_smooth = abs(minus_dm.rolling(window=period).sum())

    plus_di = 100 * (plus_dm_smooth / tr_smooth)
    minus_di = 100 * (minus_dm_smooth / tr_smooth)

    dx = (abs(plus_di - minus_di) / (plus_di + minus_di))
    adx = dx.rolling(window=period).mean()

    return plus_di, minus_di, adx

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/analyze', methods=['POST'])
def analyze():
    ticker = request.form['ticker']
    ticker = format_ticker(ticker)

    end_date = datetime.now()
    start_date = end_date - timedelta(days=200)

    data = load_stock_data(ticker, start_date.strftime('%Y-%m-%d'), end_date.strftime('%Y-%m-%d'))

    if data.empty:
```

```

        return render_template('index.html', error="No data")

    # Apply DMI and ADX calculation
    data['plus_di'], data['minus_di'], data['adx'] = calculateDMIADX(data)

    # Calculate Bostian's IIX
    data['bostian_iix'] = (data['Volume'] * ((data['Close'] - data['Open']) / data['Close']))

    # Volatility Analysis (Bollinger Bandwidth)
    length = 20
    source = data['Close']
    basis = source.rolling(window=length).mean()
    dev = source.rolling(window=length).std()
    upper = basis + 2 * dev
    lower = basis - 2 * dev
    data['bandwidth'] = (upper - lower) / basis * 100

    # Historical Mean and Standard Deviation
    data['historical_mean'] = data['bandwidth'].rolling(window=length).mean()
    data['historical_std'] = data['bandwidth'].rolling(window=length).std()

    # Percentile Calculation
    data['percentile'] = data['bandwidth'].rolling(window=length).percentile(50)

    # Regime Classification
    data['regime'] = np.where(data['bandwidth'] > (data['historical_mean'] + data['historical_std']), 'Up', np.where(data['bandwidth'] < (data['historical_mean'] - data['historical_std']), 'Down', 'Neutral'))

    # Plot Results
    plt.figure(figsize=(14, 12))

    # Plot Closing Price
    plt.subplot(4, 1, 1)
    plt.plot(data['Date'], data['Close'], label='Close Price')
    plt.title('Stock Price')
    plt.legend()

    # Plot Bollinger Bandwidth
    plt.subplot(4, 1, 2)
    plt.plot(data['Date'], data['bandwidth'], label='Bandwidth')
    plt.plot(data['Date'], data['historical_mean'], label='Historical Mean')
    plt.plot(data['Date'], data['historical_mean'] + data['historical_std'], label='Upper Band')
    plt.plot(data['Date'], data['historical_mean'] - data['historical_std'], label='Lower Band')
    plt.title('Volatility Analysis (Bandwidth)')
    plt.legend()

    # Plot Bostian's IIX
    plt.subplot(4, 1, 3)
    plt.plot(data['Date'], data['bostian_iix'], label='Bostian's IIX')
    plt.fill_between(data['Date'], data['bostian_iix'], 0, label='Bostian's IIX Accumulated')
    plt.title("Bostian's IIX Accumulated")
    plt.legend()

    # Plot Percentile of Bandwidth
    plt.subplot(4, 1, 4)
    plt.plot(data['Date'], data['percentile'], label='Bandwidth Percentile')
    plt.axhline(50, color='gray', linestyle='--', label='50th Percentile')
    plt.title('Percentile of Bandwidth')
    plt.legend()

    plt.tight_layout()

    # Save plot to a PNG image and encode it to base64
    img = io.BytesIO()
    plt.savefig(img, format='png')
    img.seek(0)
    plot_url = base64.b64encode(img.getvalue()).decode()
    plt.close()

    return render_template('result.html', plot_url=plot_url)

@app.route('/favicon.ico')
def favicon():
    return '', 204

if __name__ == "__main__":
    port = int(os.environ.get("PORT", 5000))
    app.run(host="0.0.0.0", port=port, debug=False)

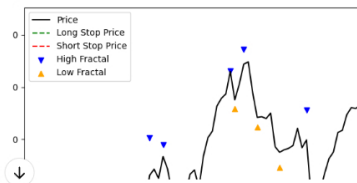
```

이 코드를 보면 한국과 미국의 주식, etf, Index 데이터를 yFinance에서 불러오는 backend code야. 우리가 만든 Python Trailing Stop Script 에 '한국과 미국의 주식, etf, Index data를 yFinance에서 불러오는 backend code '로 수정해 줄 수 있어?

🔗 Edited Python Trailing Stop Script

코드를 수정하여 한국과 미국의 주식, ETF, 인덱스 데이터를 Yahoo Finance에서 불러오는 기능을 추가했습니다. 이제 사용자가 입력한 티커를 이용해 데이터를 불러오고, 트레이딩 스탱 계산을 수행하도록 변경되었습니다.

주가 수정이나 궁금한 점이 있다면 언제든지 말씀해 주세요!



very good, but long stop price와 short stop price가 왜 안 그려졌을까?

Long Stop Price 와 Short Stop Price 가 그려지지 않은 이유는 두 가지 정도로 추측할 수 있어요:

- 결과 데이터가 NaN인 경우:** update\_trailing\_stop 함수나 flip\_trend 함수에서 계산된 값들이 NaN 일 가능성이 있습니다. 트레일링 스탱이 초기화되지 않거나 NaN으로 남아있으면 그려지지 않습니다. 각 계산 단계에서 NaN을 출력하지 않도록 초기값 설정이나 계산 과정에서의 NaN 처리 방식에 문제가 있는지 확인이 필요합니다.
- 차트 범위가 데이터 범위를 벗어난 경우:** 데이터를 불러올 때 Long\_Trail\_Output 이나 Short\_Trail\_Output 에 해당하는 값들이 NaN으로 끝까지 유지된다면, 차트에 그릴 값이 없게 됩니다. 이를 확인하기 위해 Long\_Trail\_Output 과 Short\_Trail\_Output 에 실제 값들이 제대로 존재하는지 확인해 보는 것이 필요합니다.

이 문제를 해결하기 위해 update\_trailing\_stop 과 flip\_trend 함수가 올바른 값을 계산하고 있는지 확인하고, 데이터에 NaN이 많는지 체크해 보겠습니다.

아래와 같이 수정할 수 있습니다:

- update\_trailing\_stop 함수에서 초기 값을 제대로 설정하여 첫 번째 트레일링 스탱이 NaN으로 나오지 않도록 합니다.
- flip\_trend 함수의 논리 수정으로 트레일 값이 전부 NaN으로 설정되지 않도록 합니다.

아래 코드에서 이를 수정해보겠습니다.

```

</> Python Trailing Stop Script
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import yfinance as yf
5 from datetime import datetime, timedelta
6
7 # Function to load data from Yahoo Finance
8 def load_stock_data(ticker, start_date, end_date):
9     stock = yf.download(ticker, start=start_date, end=end_date)
10    stock.reset_index(inplace=True)
11    return stock
12
13 # Function to format the ticker for Yahoo Finance
14 def format_ticker(ticker):
15     if ticker.isdigit(): # Korean stock ticker
16         return ticker + ".KS"
  
```

Edited Python Trailing Stop Script

코드를 수정하여 Long Stop Price 와 Short Stop Price 가 NaN으로 남아있지 않도록 초기 값을 설정하고 NaN 값을 처리하도록 했습니다. 이제 트레일링 스탱이 차트에 제대로 표시될 가능성이 높아졌습니다.

다시 시도해 보시고, 그래프에 값이 잘 그려졌는지 확인해 주세요. 문제가 계속된다면 추가로 도와드리겠습니다!

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, ir
  <title>Stock Analysis</title>
  <link rel="stylesheet" href="https://cdn.jsdelivrivr.net
</head>
<body>
  <div class="container mt-5">
    <h1 class="text-center">Stock, ETF, and Index An
    <form action="/analyze" method="POST" class="mt-4
      <div class="mb-3">
        <label for="ticker" class="form-label">Er
        <input type="text" id="ticker" name="tic
      </div>
      <button type="submit" class="btn btn-primary'
    </form>
    <div class="mt-5 text-center">
      <h3>Other Tools</h3>
      <div class="d-grid gap-3 d-md-flex justify-cc
        <a href="https://dmi-analysis-app-124-20:
        <a href="https://stocksanalysis-49edb39e1e
        <a href="https://bbhr-12e553aa0a18.heroki
      </div>
  
```



```

</div>
</body>
</html>

```

와

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, ir
  <title>Analysis Result</title>
  <link rel="stylesheet" href="https://cdn.jsdelivrivr.net
</head>
<body>
  <div class="container mt-5">
    <h1 class="text-center">Analysis Result</h1>
    <div class="text-center mt-4">
      
      <a href="/" class="btn btn-primary">Analyze /
    </div>
  </div>
</body>
</html>

```

은 바로 전에 참조한 backend code의 front Index.html이야,  
Python Trailing Stop Script o의 frontend Index.html 도 잘 만들어  
줄 수있어?

```

</ Trailing Stop Frontend
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Trailing Stop Analysis</title>
7   <link rel="stylesheet" href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.3.0-alpha
8 </head>
9 <body>
10 <div class="container mt-5">
11   <h1 class="text-center">Trailing Stop Analysis for Stocks, ETFs, and Indic
12   <form action="/analyze" method="POST" class="mt-4">
13     <div class="mb-3">
14       <label for="ticker" class="form-label">Enter Stock/ETF/Index Ticke
15       <input type="text" id="ticker" name="ticker" class="form-control" |
16     </div>

```

위 HTML 코드는 Python Trailing Stop Script 에 맞춰서 프론트엔드 품을 만들었습니다. 사용자가  
주식, ETF, 또는 인덱스 티커를 입력하고, 분석 결과를 얻을 수 있도록 구성되어 있습니다.  
혹시 추가 수정이 필요하거나 다른 기능을 더 원하신다면 언제든지 말씀해 주세요!

이 frontend 와 backend code를 web service를 위해 Heroku 에  
deploy 할 수 있는 코드로 만들어줘

```

</ Heroku Deploy Setup

```

위 코드와 설정 파일들은 Heroku에 Flask 애플리케이션을 배포하기 위한 백엔드 코드와 필요한 설  
정 파일들입니다. Procfile, requirements.txt 파일을 추가하여 Heroku에서 웹 서비스를 실행할  
수 있도록 구성했습니다.

### 주요 파일 구성:

- main.py (backend code): Flask 기반 웹 애플리케이션 코드.
- Procfile: Heroku에 웹 애플리케이션이 실행될 수 있도록 명령어 지정.
- requirements.txt: 필요한 라이브러리 목록 (Flask, pandas, numpy, matplotlib, yfinance, gunicorn).

위 파일들을 사용해 Heroku에 배포할 수 있습니다. 추가로 설정이나 궁금한 점이 있으시면 언제든  
지 알려주세요!

```

c:\_Fractal Trail\app.py:115: UserWarning: Starting a Mat
plt.figure(figsize=(14, 7))
[2024-11-30 19:51:14,771] ERROR in app: Exception on /ana

jinja2.exceptions.TemplateNotFound: result.html
127.0.0.1 - - [30/Nov/2024 19:51:14] "POST /analyze HTTP,

```

오류를 해결하기 위해 아래와 같은 변경 사항으 제안드립니다.

1. result.html 파일이 없음: result.html 템플릿 파일이 없어서  
jinja2.exceptions.TemplateNotFound 오류가 발생합니다. result.html 파일을 생성해서 반환

할 수 있도록 추가하겠습니다.

- 2. **Matplotlib GUI 경고:** Matplotlib이 GUI를 사용하려고 시도할 때 발생하는 경고를 피하기 위해 백엔드를 `Agg` 로 설정해 GUI를 사용하지 않도록 합니다.

아래와 같이 수정하겠습니다.

```
</> Heroku Deploy Setup
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import yfinance as yf
5 from flask import Flask, request, render_template, redirect, url_for
6 from datetime import datetime, timedelta
7 import io
8 import base64
9 import os
10
11 # Use a non-GUI backend for Matplotlib
12 plt.switch_backend('Agg')
13
14 # Create Flask app
15 app = Flask(__name__)
16
```

Edited Heroku Deploy Setup

위 코드를 수정하여 `result.html` 템플릿이 `Agg` 는 오류를 해결하고, Matplotlib의 GUI 경고 문제를 피하기 위해 GUI 백엔드를 비활성화했습니다. 이제 Flask 애플리케이션이 올바르게 작동할 가능성이 높아졌습니다.

추가적으로 `result.html` 파일이 필요하므로, 이 파일을 생성해야 합니다. 혹시 `result.html` 파일의 코드를 작성해드리길 원하시면 말씀해주세요!

result.html 파일의 코드를 작성해드리길 원합니다

```
</> Result Html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Analysis Result</title>
7 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha.
8 </head>
9 <body>
10 <div class="container mt-5">
11 <h1 class="text-center">Analysis Result</h1>
12 <div class="text-center mt-4">
13 
14 </div>
15 <div class="text-center mt-4">
16 <a href="/" class="btn btn-primary">Analyze Another Ticker</a>

```

`result.html` 파일을 생성했습니다. 이 파일은 `↓` 석 결과를 시각화한 이미지를 보여주며, 사용자가 다시 티커를 입력하여 분석할 수 있도록 링크도 포함되어 있습니다.

추가적으로 수정이나 다른 기능이 필요하시면 언제든지 말씀해 주세요!

requirements.txt: 어떻게 구성할까?

```
</> Requirements Txt
```

`requirements.txt` 파일을 생성했습니다. 이 파일에는 Heroku 배포 시 필요한 Python 라이브러리와 그 버전을 명시해 두었습니다. 필요한 라이브러리를 Heroku가 설치할 수 있도록 합니다.

추가로 도움이 필요하시면 언제든지 말씀해 주세요!

```
remote: ! Currently the root directory of your app (
remote: !
remote: ! app.py
remote: ! apppy_good.py
remote: ! Procfile
remote: ! Requirements.txt
remote: ! runtime.txt
remote: ! templates/
```

Requirements.txt `↓` requirements.txt 으로 수정해서 올리려는데 어떻게 해?

Requirements Txt 편집 실패

`requirements.txt` 파일의 이름을 수정하여 올리려면, 로컬 파일 시스템에서 해당 파일의 이름을 변경한 후 다시 커밋하고 푸시해야 합니다. 아래는 터미널에서 수행할 수 있는 절차입니다:

- 1. **파일 이름 변경:**

```
bash 코드 복사
```

```
mv Requirements.txt requirements.txt
```

## 2. Git에 변경 사항 추가:

```
bash
git add requirements.txt
git rm Requirements.txt
```

## 3. 커밋:

```
bash
git commit -m "Rename Requirements.txt to requirements.txt"
```

## 4. 변경 사항 푸시:

```
bash
git push heroku main
```

이 과정을 통해 Heroku가 올바른 파일(requirements.txt)을 인식하도록 수정할 수 있습니다.

```
2024-11-30T11:22:08.741869+00:00 app[web.1]: File
"/app/heroku/python/lib/python3.9/site-packages/flask/helpers.py",
line 14, in <module>
2024-11-30T11:22:08.741869+00:00 app[web.1]: from werkzeug.urls
import url_quote
2024-11-30T11:22:08.741870+00:00 app[web.1]: ImportError: cannot
import name 'url_quote' from 'werkzeug.urls'
(/app/heroku/python/lib/python3.9/site-packages/werkzeug/urls.py)
```

이 오류는 Flask와 Werkzeug의 버전 호환 문제로 발생하는 것 같습니다. Flask 버전을 2.0.3으로 낮추고, Werkzeug의 호환 가능한 버전을 명시적으로 지정해보겠습니다.

아래와 같이 requirements.txt 파일을 수정해 보겠습니다:

```
</> Requirements Txt
1 Flask==2.0.3
2 Werkzeug==2.0.3
3 pandas==1.4.3
4 numpy==1.23.1
5 matplotlib==3.5.2
6 yfinance==0.1.70
7 gunicorn==20.1.0
8 Jinja2==3.1.2
9
```

Edited Requirements Txt

requirements.txt 파일을 수정하여 Flask 버전을 2.0.3으로 낮추고 Werkzeug 버전을 명시했습니다. 이를 통해 호환성 문제를 해결할 수 있을 것으로 기대됩니다.

다시 Heroku에 배포하고 문제가 해결되는지 확인해 주세요. 추가적으로 도움이 필요하면 말씀해 주세요!

heroku 에 이거 수정했다고 처리하는 명령어는?

수정된 requirements.txt 파일을 Heroku에 반영하려면, 로컬 파일에서 변경한 후 다시 푸시해야 합니다. 터미널에서 다음 명령어들을 사용하세요:

## 1. 변경 사항을 Git에 추가:

```
bash
git add requirements.txt
```

## 2. 변경 사항을 커밋:

```
bash
git commit -m "Update Flask and Werkzeug versions for compatibility"
```

## 3. Heroku로 푸시:

```
bash
git push heroku main
```

이렇게 하면 수정된 requirements.txt가 Heroku에 반영됩니다.

```
PS C:\Fractal Trail> heroku logs --tail
2024-11-30T11:28:52.092068+00:00 app[web.1]: File "/app/ε
2024-11-30T11:28:52.092068+00:00 app[web.1]: return rende
2024-11-30T11:28:52.092069+00:00 app[web.1]: File "/app/.
2024-11-30T11:28:52.092069+00:00 app[web.1]: ctx.app.jin:
2024-11-30T11:28:52.092069+00:00 app[web.1]: File "/app/.
2024-11-30T11:28:52.092072+00:00 app[web.1]: return self.
2024-11-30T11:28:52.092072+00:00 app[web.1]: File "/app/ε
2024-11-30T11:28:52.092072+00:00 app[web.1]: return self.
2024-11-30T11:28:52.092072+00:00 app[web.1]: File "/app/.
2024-11-30T11:28:52.092073+00:00 app[web.1]: template = ε
2024-11-30T11:28:52.092073+00:00 app[web.1]: File "/app/.
2024-11-30T11:28:52.092073+00:00 app[web.1]: source, file
2024-11-30T11:28:52.092073+00:00 app[web.1]: File "/app/.
2024-11-30T11:28:52.092073+00:00 app[web.1]: return self.
2024-11-30T11:28:52.092073+00:00 app[web.1]: File "/app/.
2024-11-30T11:28:52.092074+00:00 app[web.1]: raise Templa
2024-11-30T11:28:52.092074+00:00 app[web.1]: jinja2.excep
2024-11-30T11:28:52.092612+00:00 app[web.1]: 10.1.50.30 -
2024-11-30T11:28:52.092872+00:00 heroku[router]: at=info
2024-11-30T11:32:11.000000+00:00 app[api]: Build started
2024-11-30T11:32:48.234006+00:00 app[api]: Deploy 9e7211:
2024-11-30T11:32:48.234006+00:00 app[api]: Release v5 cre
2024-11-30T11:32:48.596014+00:00 heroku[web.1]: Restartir
2024-11-30T11:32:48.599491+00:00 heroku[web.1]: State cha
2024-11-30T11:32:49.197875+00:00 heroku[web.1]: Stopping
2024-11-30T11:32:49.220576+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:49.220580+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:49.220585+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:49.421128+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:49.479267+00:00 heroku[web.1]: Process ε
2024-11-30T11:32:52.243514+00:00 heroku[web.1]: Starting
2024-11-30T11:32:52.887619+00:00 app[web.1]: Python builc
2024-11-30T11:32:52.887700+00:00 app[web.1]: Python builc
2024-11-30T11:32:53.147861+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:53.148299+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:53.148302+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:53.150140+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:53.192883+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:53.818582+00:00 heroku[web.1]: State cha
2024-11-30T11:33:05.000000+00:00 app[api]: Build succede
2024-11-30T11:33:12.523355+00:00 app[web.1]: [2024-11-30
2024-11-30T11:33:12.523369+00:00 app[web.1]: Traceback (n
2024-11-30T11:33:12.523371+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523372+00:00 app[web.1]: response = ε
2024-11-30T11:33:12.523372+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523372+00:00 app[web.1]: rv = self.ha
2024-11-30T11:33:12.523373+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523373+00:00 app[web.1]: rv = self.di
2024-11-30T11:33:12.523373+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523374+00:00 app[web.1]: return self.
2024-11-30T11:33:12.523374+00:00 app[web.1]: File "/app/ε
2024-11-30T11:33:12.523374+00:00 app[web.1]: return rende
2024-11-30T11:33:12.523375+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523376+00:00 app[web.1]: ctx.app.jin:
2024-11-30T11:33:12.523377+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523377+00:00 app[web.1]: return self.
2024-11-30T11:33:12.523377+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523378+00:00 app[web.1]: return self.
2024-11-30T11:33:12.523378+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523378+00:00 app[web.1]: template = ε
2024-11-30T11:33:12.523378+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523379+00:00 app[web.1]: source, file
2024-11-30T11:33:12.523379+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523379+00:00 app[web.1]: return self.
2024-11-30T11:33:12.523379+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523379+00:00 app[web.1]: raise Templa
2024-11-30T11:33:12.523379+00:00 app[web.1]: jinja2.excep
2024-11-30T11:33:12.550775+00:00 app[web.1]: 10.1.44.209
2024-11-30T11:33:12.551124+00:00 heroku[router]: at=info
2024-11-30T11:33:12.786318+00:00 app[web.1]: 10.1.44.209
2024-11-30T11:33:12.786548+00:00 heroku[router]: at=info
2024-11-30T11:33:31.482980+00:00 app[web.1]: [2024-11-30
2024-11-30T11:33:31.482990+00:00 app[web.1]: Traceback (n
2024-11-30T11:33:31.482991+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.482991+00:00 app[web.1]: response = ε
2024-11-30T11:33:31.482992+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.482992+00:00 app[web.1]: rv = self.ha
2024-11-30T11:33:31.482993+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.482993+00:00 app[web.1]: rv = self.di
2024-11-30T11:33:31.482993+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.482994+00:00 app[web.1]: return self.
2024-11-30T11:33:31.482994+00:00 app[web.1]: File "/app/ε
2024-11-30T11:33:31.482994+00:00 app[web.1]: return rende
2024-11-30T11:33:31.482994+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.483001+00:00 app[web.1]: ctx.app.jin:
2024-11-30T11:33:31.483001+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.483002+00:00 app[web.1]: return self.
2024-11-30T11:33:31.483002+00:00 app[web.1]: File "/app/.
```

```

2024-11-30T11:33:31.483002+00:00 app[web.1]: return self.
2024-11-30T11:33:31.483003+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.483003+00:00 app[web.1]: template =
2024-11-30T11:33:31.483003+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.483004+00:00 app[web.1]: source, file
2024-11-30T11:33:31.483004+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.483004+00:00 app[web.1]: return self.
2024-11-30T11:33:31.483005+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.483005+00:00 app[web.1]: raise Templa
2024-11-30T11:33:31.483005+00:00 app[web.1]: jinja2.excep
2024-11-30T11:33:31.517089+00:00 app[web.1]: 10.1.44.209
2024-11-30T11:33:31.517299+00:00 heroku[router]: at=info

```

오류 로그를 보면 `index.html` 템플릿 파일을 찾을 수 없다고 나옵니다. 이를 해결하기 위해 다음을 확인하고 처리해주세요:

#### 1. 템플릿 파일 경로 확인:

- 템플릿 파일은 Flask에서 보통 `templates/` 폴더에 있어야 합니다. `index.html` 파일이 해당 폴더에 있는지 확인하세요.
- 프로젝트 구조가 아래와 같아야 합니다:

```

markdown
├─ app.py
├─ requirements.txt
├─ Procfile
├─ templates/
│   └─ index.html
│   └─ result.html

```

#### 2. 파일을 추가하거나 수정한 경우 Git에 반영

- 파일 위치를 수정한 후 Git에 반영해야 합니다.

```

bash
git add templates/index.html templates/result.html
git commit -m "Add index.html and result.html to templates directory"
git push heroku main

```

이렇게 하면 `index.html` 과 `result.html` 파일이 Heroku 서버에 업로드되어 오류가 해결될 것입니다. 추가로 도움이 필요하시면 언제든지 말씀해 주세요!

```

PS C:\Fractal Trail> heroku logs --tail
2024-11-30T11:32:49.479267+00:00 heroku[web.1]: Process
2024-11-30T11:32:52.243514+00:00 heroku[web.1]: Starting
2024-11-30T11:32:52.887619+00:00 app[web.1]: Python build
2024-11-30T11:32:52.887700+00:00 app[web.1]: Python build
2024-11-30T11:32:53.147861+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:53.148299+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:53.148302+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:53.150140+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:53.192883+00:00 app[web.1]: [2024-11-30
2024-11-30T11:32:53.818582+00:00 heroku[web.1]: State cha
2024-11-30T11:33:05.000000+00:00 app[api]: Build succee
2024-11-30T11:33:05.252335+00:00 app[web.1]: [2024-11-30
2024-11-30T11:33:12.523369+00:00 app[web.1]: Traceback (n
2024-11-30T11:33:12.523371+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523371+00:00 app[web.1]: response =
2024-11-30T11:33:12.523372+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523372+00:00 app[web.1]: rv = self.h
2024-11-30T11:33:12.523373+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523373+00:00 app[web.1]: rv = self.d
2024-11-30T11:33:12.523373+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523374+00:00 app[web.1]: return self.
2024-11-30T11:33:12.523374+00:00 app[web.1]: File "/app/:
2024-11-30T11:33:12.523374+00:00 app[web.1]: return rende
2024-11-30T11:33:12.523375+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523376+00:00 app[web.1]: ctx.app.jin:
2024-11-30T11:33:12.523377+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523377+00:00 app[web.1]: return self.
2024-11-30T11:33:12.523377+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523378+00:00 app[web.1]: return self.
2024-11-30T11:33:12.523378+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523378+00:00 app[web.1]: template =
2024-11-30T11:33:12.523378+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523379+00:00 app[web.1]: source, file
2024-11-30T11:33:12.523379+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523379+00:00 app[web.1]: return self.
2024-11-30T11:33:12.523379+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:12.523379+00:00 app[web.1]: raise Templa
2024-11-30T11:33:12.523379+00:00 app[web.1]: jinja2.excep
2024-11-30T11:33:12.550775+00:00 app[web.1]: 10.1.44.209
2024-11-30T11:33:12.551124+00:00 heroku[router]: at=info
2024-11-30T11:33:12.786318+00:00 app[web.1]: 10.1.44.209
2024-11-30T11:33:12.786548+00:00 heroku[router]: at=info

```

```
2024-11-30T11:33:31.482980+00:00 app[web.1]: [2024-11-30
2024-11-30T11:33:31.482990+00:00 app[web.1]: Traceback (n
2024-11-30T11:33:31.482991+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.482991+00:00 app[web.1]: response = s
2024-11-30T11:33:31.482992+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.482992+00:00 app[web.1]: rv = self.ha
2024-11-30T11:33:31.482993+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.482993+00:00 app[web.1]: rv = self.di
2024-11-30T11:33:31.482993+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.482994+00:00 app[web.1]: return self.
2024-11-30T11:33:31.482994+00:00 app[web.1]: File "/app/i
2024-11-30T11:33:31.482994+00:00 app[web.1]: return rende
2024-11-30T11:33:31.482994+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.483001+00:00 app[web.1]: ctx.app.jin;
2024-11-30T11:33:31.483001+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.483002+00:00 app[web.1]: return self.
2024-11-30T11:33:31.483002+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.483002+00:00 app[web.1]: return self.
2024-11-30T11:33:31.483003+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.483003+00:00 app[web.1]: template = s
2024-11-30T11:33:31.483003+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.483004+00:00 app[web.1]: source, file
2024-11-30T11:33:31.483004+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.483004+00:00 app[web.1]: return self.
2024-11-30T11:33:31.483005+00:00 app[web.1]: File "/app/.
2024-11-30T11:33:31.483005+00:00 app[web.1]: raise Templa
2024-11-30T11:33:31.483005+00:00 app[web.1]: jinja2.excep
2024-11-30T11:33:31.517089+00:00 app[web.1]: 10.1.44.209
2024-11-30T11:33:31.517299+00:00 heroku[router]: at=info
2024-11-30T11:42:39.959829+00:00 app[web.1]: [2024-11-30
2024-11-30T11:42:39.959842+00:00 app[web.1]: Traceback (n
2024-11-30T11:42:39.959843+00:00 app[web.1]: File "/app/.
2024-11-30T11:42:39.959844+00:00 app[web.1]: response = s
2024-11-30T11:42:39.959845+00:00 app[web.1]: File "/app/.
2024-11-30T11:42:39.959845+00:00 app[web.1]: rv = self.ha
2024-11-30T11:42:39.959845+00:00 app[web.1]: File "/app/.
2024-11-30T11:42:39.959846+00:00 app[web.1]: rv = self.di
2024-11-30T11:42:39.959846+00:00 app[web.1]: File "/app/.
2024-11-30T11:42:39.959847+00:00 app[web.1]: return self.
2024-11-30T11:42:39.959847+00:00 app[web.1]: File "/app/i
2024-11-30T11:42:39.959847+00:00 app[web.1]: return rende
2024-11-30T11:42:39.959848+00:00 app[web.1]: File "/app/.
2024-11-30T11:42:39.959849+00:00 app[web.1]: ctx.app.jin;
2024-11-30T11:42:39.959850+00:00 app[web.1]: File "/app/.
2024-11-30T11:42:39.959851+00:00 app[web.1]: return self.
2024-11-30T11:42:39.959851+00:00 app[web.1]: File "/app/.
2024-11-30T11:42:39.959851+00:00 app[web.1]: return self.
2024-11-30T11:42:39.959851+00:00 app[web.1]: File "/app/.
2024-11-30T11:42:39.959852+00:00 app[web.1]: template = s
2024-11-30T11:42:39.959852+00:00 app[web.1]: File "/app/.
2024-11-30T11:42:39.959852+00:00 app[web.1]: source, file
2024-11-30T11:42:39.959853+00:00 app[web.1]: File "/app/.
2024-11-30T11:42:39.959853+00:00 app[web.1]: return self.
2024-11-30T11:42:39.959853+00:00 app[web.1]: File "/app/.
2024-11-30T11:42:39.959853+00:00 app[web.1]: raise Templa
2024-11-30T11:42:39.959854+00:00 app[web.1]: jinja2.excep
2024-11-30T11:42:39.960435+00:00 app[web.1]: 10.1.36.128
2024-11-30T11:42:39.960610+00:00 heroku[router]: at=info
```

오류 로그에 따르면 `index.html` 템플릿 파일을 찾지 못하고 있습니다. 이 문제를 해결하려면 다음을 확인하고 진행해 보세요.

1. `index.html` 파일의 위치 확인:

- `index.html` 파일은 Flask 애플리케이션에서 기본적으로 `templates` 폴더 안에 있어야 합니다. 이 파일이 제대로 위치해 있는지 확인하세요.
- 예를 들어, 프로젝트 구조는 다음과 같아야 합니다:

```
markdown
├─ app.py
├─ requirements.txt
├─ Procfile
├─ templates/
│  └─ index.html
│  └─ result.html
```

2. `templates` 폴더 추가 및 파일 확인:

- 만약 `templates` 폴더가 없다면 생성하고, 그 안에 `index.html` 및 `result.html` 파일을 추가하세요.

3. Git에 변경 사항 추가:

- 파일 위치를 수정한 후 변경 사항을 Git에 추가하고 Heroku에 다시 배포해야 합니다.

```
bash
git add templates/index.html templates/result.html
git commit -m "Add index.html and result.html to templates directory"
```

```
git push heroku main
```

이렇게 하면 `index.html` 과 `result.html` 이 Heroku에 배포되며, 템플릿을 찾을 수 없는 오류를 해결할 수 있을 것입니다. 추가로 도움이 필요하시면 언제든지 알려주세요!

```
PS C:\Fractal Trail> git commit -m "Add index.html and result.html to templates directory"
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
deleted:    apppy_good.py

no changes added to commit (use "git add" and/or "git commit -a")
```

Git에서 `apppy_good.py` 파일이 삭제된 상태인데, 아직 커밋되지 않았기 때문에 오류가 발생하고 있습니다. 삭제 상태를 포함하여 변경 사항을 커밋하거나, 해당 파일을 복원해야 합니다.

삭제된 파일을 포함해 모든 변경 사항을 커밋하려면 다음 명령어를 사용하세요:

### 1. 삭제 상태 포함하여 변경 사항 추가:

```
bash
git add -A
```

### 2. 커밋:

```
bash
git commit -m "Add index.html and result.html to templates directory, remove unused fi
```

### 3. Heroku로 푸시:

```
bash
git push heroku main
```

만약 `apppy_good.py` 파일을 복원하고 싶다면 다음 명령어를 사용하세요:

```
bash
git restore apppy_good.py
```

그 후 `git add`, `git commit`, `git push` 를 진행하시면 됩니다.

```
PS C:\Fractal Trail> heroku logs --tail
2024-11-30T11:47:52.698140+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:52.698140+00:00 app[web.1]: source, file
2024-11-30T11:47:52.698140+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:52.698141+00:00 app[web.1]: return self.
2024-11-30T11:47:52.698141+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:52.698141+00:00 app[web.1]: raise Templa
2024-11-30T11:47:52.698141+00:00 app[web.1]: jinja2.excej
2024-11-30T11:47:52.698618+00:00 app[web.1]: 10.1.47.112
2024-11-30T11:47:52.699314+00:00 heroku[router]: at=info
2024-11-30T11:47:53.182781+00:00 app[web.1]: [2024-11-30
2024-11-30T11:47:53.182794+00:00 app[web.1]: Traceback (n
2024-11-30T11:47:53.182795+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:53.182802+00:00 app[web.1]: response =
2024-11-30T11:47:53.182811+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:53.182812+00:00 app[web.1]: rv = self.ha
2024-11-30T11:47:53.182812+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:53.182812+00:00 app[web.1]: rv = self.di
2024-11-30T11:47:53.182812+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:53.182812+00:00 app[web.1]: return self.
2024-11-30T11:47:53.182813+00:00 app[web.1]: File "/app/:
2024-11-30T11:47:53.182813+00:00 app[web.1]: return rende
2024-11-30T11:47:53.182813+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:53.182814+00:00 app[web.1]: ctx.app.jin:
2024-11-30T11:47:53.182815+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:53.182816+00:00 app[web.1]: return self.
2024-11-30T11:47:53.182816+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:53.182816+00:00 app[web.1]: return self.
2024-11-30T11:47:53.182816+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:53.182816+00:00 app[web.1]: template =
2024-11-30T11:47:53.182816+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:53.182817+00:00 app[web.1]: source, file
2024-11-30T11:47:53.182817+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:53.182817+00:00 app[web.1]: return self.
2024-11-30T11:47:53.182817+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:53.182817+00:00 app[web.1]: raise Templa
2024-11-30T11:47:53.182817+00:00 app[web.1]: jinja2.excej
2024-11-30T11:47:53.183421+00:00 app[web.1]: 10.1.47.112
2024-11-30T11:47:53.183516+00:00 heroku[router]: at=info
2024-11-30T11:47:53.670551+00:00 app[web.1]: [2024-11-30
```

```

2024-11-30T11:47:59.670666+00:00 app[web.1]: [2024-11-30
2024-11-30T11:47:59.670666+00:00 app[web.1]: Traceback (n
2024-11-30T11:47:59.670667+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:59.670667+00:00 app[web.1]: response = s
2024-11-30T11:47:59.670668+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:59.670668+00:00 app[web.1]: rv = self.ha
2024-11-30T11:47:59.670668+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:59.670668+00:00 app[web.1]: rv = self.di
2024-11-30T11:47:59.670669+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:59.670669+00:00 app[web.1]: return self.
2024-11-30T11:47:59.670676+00:00 app[web.1]: File "/app/:
2024-11-30T11:47:59.670676+00:00 app[web.1]: return rende
2024-11-30T11:47:59.670676+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:59.670677+00:00 app[web.1]: ctx.app.jin:
2024-11-30T11:47:59.670678+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:59.670678+00:00 app[web.1]: return self.
2024-11-30T11:47:59.670678+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:59.670678+00:00 app[web.1]: return self.
2024-11-30T11:47:59.670679+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:59.670679+00:00 app[web.1]: template = s
2024-11-30T11:47:59.670679+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:59.670680+00:00 app[web.1]: source, file
2024-11-30T11:47:59.670680+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:59.670681+00:00 app[web.1]: return self.
2024-11-30T11:47:59.670681+00:00 app[web.1]: File "/app/.
2024-11-30T11:47:59.670681+00:00 app[web.1]: raise Templ
2024-11-30T11:47:59.670681+00:00 app[web.1]: jinjaj2.excep
2024-11-30T11:47:59.671136+00:00 app[web.1]: 10.1.47.112
2024-11-30T11:47:59.671351+00:00 heroku[router]: at=info
2024-11-30T11:47:59.935407+00:00 app[web.1]: 10.1.47.112
2024-11-30T11:47:59.935827+00:00 heroku[router]: at=info
2024-11-30T11:48:03.972369+00:00 app[web.1]: [2024-11-30
2024-11-30T11:48:03.972380+00:00 app[web.1]: Traceback (n
2024-11-30T11:48:03.972381+00:00 app[web.1]: File "/app/.
2024-11-30T11:48:03.972381+00:00 app[web.1]: response = s
2024-11-30T11:48:03.972382+00:00 app[web.1]: File "/app/.
2024-11-30T11:48:03.972382+00:00 app[web.1]: rv = self.ha
2024-11-30T11:48:03.972393+00:00 app[web.1]: File "/app/.
2024-11-30T11:48:03.972393+00:00 app[web.1]: rv = self.di
2024-11-30T11:48:03.972393+00:00 app[web.1]: File "/app/.
2024-11-30T11:48:03.972394+00:00 app[web.1]: return self.
2024-11-30T11:48:03.972394+00:00 app[web.1]: File "/app/:
2024-11-30T11:48:03.972394+00:00 app[web.1]: return rende
2024-11-30T11:48:03.972394+00:00 app[web.1]: File "/app/.
2024-11-30T11:48:03.972395+00:00 app[web.1]: ctx.app.jin:
2024-11-30T11:48:03.972395+00:00 app[web.1]: File "/app/.
2024-11-30T11:48:03.972395+00:00 app[web.1]: return self.
2024-11-30T11:48:03.972396+00:00 app[web.1]: File "/app/.
2024-11-30T11:48:03.972396+00:00 app[web.1]: return self.
2024-11-30T11:48:03.972396+00:00 app[web.1]: File "/app/.
2024-11-30T11:48:03.972396+00:00 app[web.1]: File "/app/.
2024-11-30T11:48:03.972396+00:00 app[web.1]: template = s
2024-11-30T11:48:03.972396+00:00 app[web.1]: File "/app/.
2024-11-30T11:48:03.972397+00:00 app[web.1]: source, file
2024-11-30T11:48:03.972397+00:00 app[web.1]: File "/app/.
2024-11-30T11:48:03.972397+00:00 app[web.1]: return self.
2024-11-30T11:48:03.972397+00:00 app[web.1]: File "/app/.
2024-11-30T11:48:03.972397+00:00 app[web.1]: raise Templ
2024-11-30T11:48:03.972398+00:00 app[web.1]: jinjaj2.excep
2024-11-30T11:48:03.972872+00:00 app[web.1]: 10.1.47.112
2024-11-30T11:48:03.973097+00:00 heroku[router]: at=info
2024-11-30T11:48:04.233060+00:00 app[web.1]: 10.1.47.112
2024-11-30T11:48:04.233518+00:00 heroku[router]: at=info

```

여전히 `index.html` 파일을 찾을 수 없다는 오류가 발생하고 있습니다. 이 문제는 보통 `index.html` 파일이 존재하지 않거나, 잘못된 경로에 배치된 경우에 발생합니다. 다음과 같은 방법으로 문제를 해결해 보세요:

### 1. 파일 위치 확인:

- `index.html` 파일이 `templates` 디렉터리에 있는지 확인하세요. 파일이 없다면 추가해야 합니다.
- 프로젝트 구조는 다음과 같아야 합니다:

```

markdown
├─ app.py
├─ requirements.txt
├─ Procfile
├─ templates/
│   └─ index.html
│   └─ result.html

```

### 2. 파일 추가 후 Git에 반영:

- `index.html` 파일을 `templates` 폴더에 추가한 후, 변경 사항을 Git에 반영하고 다시 무시해야 합니다.

```

bash
git add templates/index.html templates/result.html

```

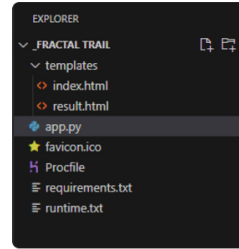


```
git commit -m "Add index.html and result.html to templates directory"
git push heroku main
```

### 3. 파일 경로 확인:

- Flask는 템플릿 파일이 반드시 `templates/` 폴더에 있어야 합니다. 파일 경로가 잘못된 경우 `index.html` 을 찾을 수 없게 되니, 반드시 올바른 경로에 위치해야 합니다.

이 과정을 통해 Heroku에서 `index.html` 을 찾을 수 있게 됩니다. 추가로 문제가 있거나 더 도와드릴 부분이 있다면 알려주세요!



프로젝트 구조를 확인한 결과, `templates` 폴더에 `index.html` 과 `result.html` 파일이 올바르게 배치되어 있는 것 같습니다. 이 경우에도 Heroku에서 템플릿을 찾지 못하는 문제가 발생하는 몇 가지 원인을 살펴보고 해결해 보겠습니다.

#### 1. Git에 올바르게 반영되지 않은 파일 확인:

- 모든 파일이 Git에 커밋되어 있는지 확인하세요. 특히, `index.html` 과 `result.html` 파일이 올바르게 Git에 추가되었는지 확인해 ↓ 합니다.

```
bash
git status
```

만약 `index.html` 파일이 Git에 추가되지 않았다면, 추가하고 푸시하세요.

```
bash
git add templates/index.html templates/result.html
git commit -m "Add index.html and result.html"
git push heroku main
```

#### 2. 템플릿 캐시 문제:

- 가끔씩 템플릿 파일이 캐시되어 발생하는 문제일 수 있습니다. Heroku에서 앱을 강제로 다시 시작해 볼 수 있습니다.

```
bash
heroku restart
```

#### 3. 경로 확인 및 대소문자 확인:

- 파일명이나 경로의 대소문자가 일치하지 않으면 템플릿을 찾지 못할 수 있습니다. `templates` 폴더와 파일 이름(`index.html`, `result.html`)의 대소문자가 정확한지 확인하세요.

#### 4. 추가 로그 확인:

- 여전히 문제가 해결되지 않는 경우, 추가 로그를 확인해 더 많은 힌트를 얻을 수 있습니다. `heroku logs --tail` 을 사용해 실시간으로 로그를 확인하면서 추가적인 오류가 있는지 찾아보세요.

이 단계를 시도한 후에도 문제가 해결되지 않으면 알려주세요. 추가적인 문제 해결을 도와드리겠습니다.

```
PS C:\Fractal Trail> git status
On branch master
nothing to commit, working tree clean
```

현재 `git status` 명령어에서 "nothing to commit, working tree clean" 메시지가 나타났으므로, 모든 변경 사항이 이미 커밋되어 있다는 의미입니다. 따라서 `index.html` 파일은 Git에 올바르게 커밋되었고, Heroku에도 푸시된 상태입니다. 그럼에도 여전히 Heroku에서 템플릿을 찾지 못하는 이유는 몇 가지 가능성이 있습니다.

#### 1. Heroku 배포 캐시 문제:

- Heroku의 캐시 때문에 변경 사항이 제대로 반영되지 않았을 수 있습니다. 이를 해결하기 위해 캐시를 무시하고 다시 배포해 볼 수 있습니다.

```
bash
git commit --allow-empty -m "Trigger Heroku redeploy"
git push heroku main
```

#### 2. 디렉터리 구조 문제:

#### 6. 디렉터리 구조 확인

- 디렉터리 구조를 다시 확인해 보세요. `templates` 폴더와 그 안의 파일들이 Heroku에서 접근할 수 있는 위치에 있는지 확인해야 합니다. 구조가 잘못되면 템플릿을 찾을 수 없습니다.

#### 3. 템플릿 파일 권한 문제:

- 파일 권한 문제로 인해 Heroku에서 템플릿을 읽지 못하는 경우도 있을 수 있습니다. 권한이 잘못 설정되었다면, 다음 명령어로 모든 파일 권한을 확인하고 수정하세요.

```
bash 🔗 코드 복사  
  
chmod -R 755 .
```

#### 4. Heroku 로그 추가 확인:

- `heroku logs --tail` 명령어로 추가적인 오류나 힌트가 있는지 확인하세요. 특정 권한 문제나 접근 문제에 대한 추가적인 로그가 있을 수 있습니다.

위의 단계를 따라 진행해보시고, 문제가 해결되는지 확인해 주세요. 추가적인 로그가 있다면 다시 공유해 주시면 더 도와드릴 수 있습니다.

```
PS C:\Fractal Trail> heroku logs --tail  
2024-11-30T11:52:43.763078+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:43.763078+00:00 app[web.1]: return self.  
2024-11-30T11:52:43.763079+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:43.763079+00:00 app[web.1]: return rende  
2024-11-30T11:52:43.763079+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:43.763081+00:00 app[web.1]: ctx.app.jin  
2024-11-30T11:52:43.763082+00:00 app[web.1]: File "/app/.  
line 1081, in get_or_select_template  
2024-11-30T11:52:43.763082+00:00 app[web.1]: return self.  
2024-11-30T11:52:43.763082+00:00 app[web.1]: File "/app/.  
line 1010, in get_template  
2024-11-30T11:52:43.763083+00:00 app[web.1]: return self.  
2024-11-30T11:52:43.763083+00:00 app[web.1]: File "/app/.  
line 969, in _load_template  
2024-11-30T11:52:43.763083+00:00 app[web.1]: template = s  
2024-11-30T11:52:43.763083+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:43.763095+00:00 app[web.1]: source, file  
2024-11-30T11:52:43.763095+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:43.763095+00:00 app[web.1]: return self.  
2024-11-30T11:52:43.763095+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:43.763095+00:00 app[web.1]: raise Templa  
2024-11-30T11:52:43.763095+00:00 app[web.1]: jinja2.except  
2024-11-30T11:52:43.763516+00:00 app[web.1]: 10.1.29.84 -  
2024-11-30T11:52:43.763887+00:00 heroku[router]: at=info  
2024-11-30T11:52:43.995783+00:00 app[web.1]: 10.1.29.84 -  
(KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"  
2024-11-30T11:52:43.996001+00:00 heroku[router]: at=info  
2024-11-30T11:52:47.338294+00:00 app[web.1]: [2024-11-30  
2024-11-30T11:52:47.338304+00:00 app[web.1]: Traceback (m  
2024-11-30T11:52:47.338307+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:47.338307+00:00 app[web.1]: response = s  
2024-11-30T11:52:47.338307+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:47.338308+00:00 app[web.1]: rv = self.h  
2024-11-30T11:52:47.338308+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:47.338308+00:00 app[web.1]: rv = self.di  
2024-11-30T11:52:47.338331+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:47.338331+00:00 app[web.1]: return self.  
2024-11-30T11:52:47.338332+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:47.338332+00:00 app[web.1]: return rende  
2024-11-30T11:52:47.338332+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:47.338333+00:00 app[web.1]: ctx.app.jin  
2024-11-30T11:52:47.338334+00:00 app[web.1]: File "/app/.  
line 1081, in get_or_select_template  
2024-11-30T11:52:47.338334+00:00 app[web.1]: return self.  
2024-11-30T11:52:47.338334+00:00 app[web.1]: File "/app/.  
line 1010, in get_template  
2024-11-30T11:52:47.338335+00:00 app[web.1]: return self.  
2024-11-30T11:52:47.338335+00:00 app[web.1]: File "/app/.  
line 969, in _load_template  
2024-11-30T11:52:47.338335+00:00 app[web.1]: template = s  
2024-11-30T11:52:47.338335+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:47.338335+00:00 app[web.1]: source, file  
2024-11-30T11:52:47.338336+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:47.338336+00:00 app[web.1]: return self.  
2024-11-30T11:52:47.338336+00:00 app[web.1]: File "/app/.  
2024-11-30T11:52:47.338336+00:00 app[web.1]: raise Templa  
2024-11-30T11:52:47.338336+00:00 app[web.1]: jinja2.except  
2024-11-30T11:52:47.365664+00:00 app[web.1]: 10.1.29.84 -  
2024-11-30T11:52:47.366372+00:00 heroku[router]: at=info  
bytes=463 protocol=https  
2024-11-30T11:52:47.666003+00:00 app[web.1]: 10.1.29.84 -  
(KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"  
2024-11-30T11:52:47.666361+00:00 heroku[router]: at=info  
2024-11-30T11:53:02.542301+00:00 heroku[web.1]: Restartin  
2024-11-30T11:53:02.591483+00:00 heroku[web.1]: State ch  
2024-11-30T11:53:03.122931+00:00 heroku[web.1]: Stopping  
2024-11-30T11:53:03.145855+00:00 app[web.1]: [2024-11-30
```

```

2024-11-30T11:53:03.145872+00:00 app[web.1]: [2024-11-30
2024-11-30T11:53:03.145885+00:00 app[web.1]: [2024-11-30
2024-11-30T11:53:03.346366+00:00 app[web.1]: [2024-11-30
2024-11-30T11:53:03.402634+00:00 heroku[web.1]: Process €
2024-11-30T11:53:06.708741+00:00 heroku[web.1]: Starting
2024-11-30T11:53:07.443968+00:00 app[web.1]: Python builc
2024-11-30T11:53:07.443980+00:00 app[web.1]: Python builc
2024-11-30T11:53:07.755658+00:00 app[web.1]: [2024-11-30
2024-11-30T11:53:07.755997+00:00 app[web.1]: [2024-11-30
2024-11-30T11:53:07.756028+00:00 app[web.1]: [2024-11-30
2024-11-30T11:53:07.758632+00:00 app[web.1]: [2024-11-30
2024-11-30T11:53:07.828025+00:00 heroku[web.1]: State ch
2024-11-30T11:53:07.850224+00:00 app[web.1]: [2024-11-30
2024-11-30T11:53:50.276742+00:00 app[web.1]: [2024-11-30
2024-11-30T11:53:50.276756+00:00 app[web.1]: Traceback (n
2024-11-30T11:53:50.276757+00:00 app[web.1]: File "/app/.
2024-11-30T11:53:50.276758+00:00 app[web.1]: response = s
2024-11-30T11:53:50.276759+00:00 app[web.1]: File "/app/.
2024-11-30T11:53:50.276760+00:00 app[web.1]: rv = self.ha
2024-11-30T11:53:50.276760+00:00 app[web.1]: File "/app/.
2024-11-30T11:53:50.276761+00:00 app[web.1]: rv = self.di
2024-11-30T11:53:50.276761+00:00 app[web.1]: File "/app/.
2024-11-30T11:53:50.276761+00:00 app[web.1]: return self.
2024-11-30T11:53:50.276761+00:00 app[web.1]: File "/app/:
2024-11-30T11:53:50.276762+00:00 app[web.1]: return rende
2024-11-30T11:53:50.276762+00:00 app[web.1]: File "/app/.
2024-11-30T11:53:50.276763+00:00 app[web.1]: ctx.app.jin;
2024-11-30T11:53:50.276764+00:00 app[web.1]: File "/app/.
2024-11-30T11:53:50.276765+00:00 app[web.1]: return self.
2024-11-30T11:53:50.276765+00:00 app[web.1]: return self.
2024-11-30T11:53:50.276765+00:00 app[web.1]: File "/app/.
2024-11-30T11:53:50.276766+00:00 app[web.1]: template = s
2024-11-30T11:53:50.276766+00:00 app[web.1]: File "/app/.
2024-11-30T11:53:50.276766+00:00 app[web.1]: source, file
2024-11-30T11:53:50.276766+00:00 app[web.1]: File "/app/.
2024-11-30T11:53:50.276767+00:00 app[web.1]: return self.
2024-11-30T11:53:50.276767+00:00 app[web.1]: File "/app/.
2024-11-30T11:53:50.276767+00:00 app[web.1]: raise Templa
2024-11-30T11:53:50.276767+00:00 app[web.1]: jinja2.excep
2024-11-30T11:53:50.323199+00:00 app[web.1]: 10.1.86.240
2024-11-30T11:53:50.323227+00:00 heroku[router]: at=info

```

Heroku 로그에서 `jinja2.exceptions.TemplateNotFound: index.html` 오류가 계속 발생하고 있습니다. 이 문제의 해결을 위해 다음을 확인해보세요:

1. **templates** 디렉터리의 위치 확인:

- `index.html` 파일이 `templates` 디렉터리 안에 있는지 다시 한 번 확인해주세요. 디렉터리 구조가 정확해야 Flask가 템플릿을 찾을 수 있습니다.

2. **Git 상태 확인 후 다시 푸시:**

- `templates/index.html` 이 Git에서 추적되고 있는지 확인하고, Heroku에 푸시하는 과정에서 파일이 누락되지 않도록 해야 합니다.
- 다음 명령어를 통해 `templates/index.html` 을 Git에 추가하고 커밋한 뒤, Heroku에 다시 푸시해보세요.

```

bash
git add templates/index.html templates/result.html
git commit -m "Add index.html and result.html to templates directory"
git push heroku master

```

3. **템플릿 캐시 문제:**

- 가끔 Heroku 배포에서 캐시 문제로 인해 파일 업데이트가 반영되지 않을 수 있습니다. 캐시를 무시하고 다시 배포하도록 아래 명령어를 사용할 수 있습니다.

```

bash
git commit --allow-empty -m "Trigger Heroku redeploy"
git push heroku master

```

이 단계를 수행한 후 다시 Heroku 로그를 확인(↓)하세요. 여전히 문제가 있다면 추가적인 로그를 공유해주시면 더 도와드릴 수 있습니다.

```

2024-11-30T11:58:02.774967+00:00 app[web.1]: File "/app/.
2024-11-30T11:58:02.774967+00:00 app[web.1]: template = s
2024-11-30T11:58:02.774967+00:00 app[web.1]: File "/app/.
2024-11-30T11:58:02.774967+00:00 app[web.1]: source, file
2024-11-30T11:58:02.774968+00:00 app[web.1]: File "/app/.
2024-11-30T11:58:02.774968+00:00 app[web.1]: return self.
2024-11-30T11:58:02.774968+00:00 app[web.1]: File "/app/.
2024-11-30T11:58:02.774968+00:00 app[web.1]: raise Templa
2024-11-30T11:58:02.774968+00:00 app[web.1]: jinja2.excep
2024-11-30T11:58:02.811532+00:00 app[web.1]: 10.1.36.128
2024-11-30T11:58:02.811658+00:00 heroku[router]: at=info

```



Heroku 로그에서 `jinja2.exceptions.TemplateNotFound: index.html` 오류가 발생하고 있습니다. 아래 단계를 통해 문제를 해결해보세요:

1. `index.html` 파일 위치 확인:

- `index.html` 파일이 반드시 프로젝트 `templates` 폴더 내에 있어야 합니다.
- 현재 `templates` 폴더가 루트에 있고 그 안에 `index.html` 파일이 있는지 다시 확인하세요.

2. Git 추적 상태 확인:

- Heroku에서 배포된 프로젝트에 `index.html` 이 제대로 포함되지 않았을 수 있습니다. 다음 명령어로 해당 파일을 Git에 추가하고 다시 푸시하세요:

```
bash
git add templates/index.html templates/result.html
git commit -m "Add missing template files"
git push heroku master
```

3. 템플릿 경로가 정확한지 확인:

- Flask에서 템플릿을 렌더링할 때는 기본적으로 `templates` 디렉터리를 찾습니다. 따라서 코드에서 `render_template('index.html')` 라고만 사용하면 됩니다.
- `app.py` 코드에서 `render_template()` 함수 호출이 제대로 되어 있는지 확인해주세요.

4. Heroku 재배포:

- Heroku에서 배포 시에 캐시 문제로 인해 파일이 적용되지 않을 수 있습니다. 새로운 변경 사항을 확실히 반영하기 위해서 빈 커밋을 만들어 다시 배포할 수 있습니다:

```
bash
git commit --allow-empty -m "Trigger redeploy"
git push heroku master
```

이 단계를 모두 수행한 후 Heroku 로그를 다시 확인하고 문제가 해결되는지 확인해보세요.

app.py 코드에서 `render_template()` 함수 호출이 제대로 '안되어' 있어요

`app.py`에서 `render_template()` 함수 호출을 제대로 고치기 위해서, 다음 사항을 확인하세요:

1. 템플릿 파일 이름과 경로 확인:

- 템플릿 파일(`index.html`)이 `templates` 폴더 안에 있어야 합니다.
- `render_template()` 함수에 전달되는 파일 이름이 정확하게 `index.html` 로 되어 있는지 확인하세요.

2. `render_template()` 함수의 사용 예시:

- `render_template()` 함수 호출 부분이 제대로 작성되었는지 확인해야 합니다. 예를 들어:

```
python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)
```

3. 확인 후 적용:

- 위 코드를 확인하여 `render_template('index.html')` 함수 호출이 정확하게 작성되었는지, Flask의 `templates` 디렉터리에서 해당 파일을 찾을 수 있는지 확인해주세요.

혹시 수정한 부분을 보여주시면, 구체적으로 어떤 부분에서 문제가 있었는지 더 도와드릴 수 있어요.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
from flask import Flask, request, render_template, redirect
from datetime import datetime, timedelta
import io
import base64
import os

# Use a non-GUI backend for Matplotlib
plt.switch_backend('Agg')

# Create Flask app
app = Flask(__name__)
```

```

# Function to load data from Yahoo Finance
def load_stock_data(ticker, start_date, end_date):
    stock = yf.download(ticker, start=start_date, end=end_date)
    stock.reset_index(inplace=True)
    return stock

# Function to format the ticker for Yahoo Finance
def format_ticker(ticker):
    if ticker.isdigit(): # Korean stock ticker
        return ticker + ".KS"
    return ticker # US or other tickers

# Route for the index page
@app.route('/')
def index():
    return render_template('index.html')

# Route for analyzing the stock data
@app.route('/analyze', methods=['POST'])
def analyze():
    ticker = request.form['ticker']
    ticker = format_ticker(ticker)

    end_date = datetime.now()
    start_date = end_date - timedelta(days=200)

    df = load_stock_data(ticker, start_date.strftime('%Y-%m-%d'), end_date.strftime('%Y-%m-%d'))

    if df.empty:
        return render_template('index.html', error="No data found")

    # 1. Find Fractals
    def find_fractals(df, left_range=2, right_range=2):
        high_fractals = []
        low_fractals = []

        for i in range(left_range, len(df) - right_range):
            is_high_fractal = all(df['High'][i:i+left_range] > df['High'][i+left_range:i+left_range+right_range])
            is_low_fractal = all(df['Low'][i:i+left_range] < df['Low'][i+left_range:i+left_range+right_range])

            high_fractals.append(df['High'][i] if is_high_fractal else None)
            low_fractals.append(df['Low'][i] if is_low_fractal else None)

        high_fractals = [np.nan] * left_range + high_fractals
        low_fractals = [np.nan] * left_range + low_fractals

        return pd.Series(high_fractals, index=df.index), pd.Series(low_fractals, index=df.index)

    df['High_Fractal'], df['Low_Fractal'] = find_fractals(df)

    # 2. Add Buffer to Fractals
    def add_buffer(fractal_prices, buffer_percent, direction):
        if direction == 'plus':
            return fractal_prices * (1 + buffer_percent / 100)
        elif direction == 'minus':
            return fractal_prices * (1 - buffer_percent / 100)
        else:
            return fractal_prices

    df['High_Fractal_Buffered'] = add_buffer(df['High_Fractal'], 5, 'plus')
    df['Low_Fractal_Buffered'] = add_buffer(df['Low_Fractal'], 5, 'minus')

    # 3. Update Trailing Stop
    def update_trailing_stop(fractal_prices, previous_trail):
        trail = []
        for i in range(len(fractal_prices)):
            if i == 0:
                trail.append(fractal_prices[i] if not np.isnan(fractal_prices[i]) else None)
            else:
                if direction == 'down' and price_data[i] < previous_trail:
                    trail.append(previous_trail)
                elif direction == 'up' and price_data[i] > previous_trail:
                    trail.append(previous_trail)
                else:
                    trail.append(fractal_prices[i] if not np.isnan(fractal_prices[i]) else None)
            previous_trail = trail[-1]
        return pd.Series(trail, index=fractal_prices.index)

    df['Long_Trailing_Stop'] = update_trailing_stop(df['High_Fractal_Buffered'], df['High_Fractal_Buffered'].iloc[0])
    df['Short_Trailing_Stop'] = update_trailing_stop(df['Low_Fractal_Buffered'], df['Low_Fractal_Buffered'].iloc[0])

    # 4. Flip Trend Logic
    def flip_trend(price_data, long_trail, short_trail):
        is_long = True
        long_trail_output = []
        short_trail_output = []

        for i in range(len(price_data)):
            if is_long and price_data[i] < short_trail[i]:
                is_long = False
            elif not is_long and price_data[i] > long_trail[i]:
                is_long = True
            long_trail_output.append(long_trail[i])
            short_trail_output.append(short_trail[i])
    
```

```

        is_long = True

        long_trail_output.append(long_trail[i] if is_
short_trail_output.append(short_trail[i] if r

    return pd.Series(long_trail_output, index=price_c

df['Long_Trail_Output'], df['Short_Trail_Output'] = f

# Plotting the Results
plt.figure(figsize=(14, 7))
plt.plot(df.index, df['Close'], label='Price', color=
plt.plot(df.index, df['Long_Trail_Output'], label='Lc
plt.plot(df.index, df['Short_Trail_Output'], label='S
plt.scatter(df.index, df['High_Fractal'], label='High
plt.scatter(df.index, df['Low_Fractal'], label='Low f
plt.legend()
plt.tight_layout()

# Save plot to a PNG image and encode it to base64
img = io.BytesIO()
plt.savefig(img, format='png')
img.seek(0)
plot_url = base64.b64encode(img.getvalue()).decode()
plt.close()

return render_template('result.html', plot_url=plot_u

if __name__ == "__main__":
    port = int(os.environ.get("PORT", 5000))
    app.run(host="0.0.0.0", port=port, debug=False)

# Procfile
# web: python main.py

# requirements.txt
# Flask
# pandas
# numpy
# matplotlib
# yfinance
# gunicorn

```

To address the `render_template()` error (`TemplateNotFound: index.html`), you should check the following:

#### 1. Ensure the Correct Directory Structure ↓

- Your `templates` folder should be in the same directory as `app.py`.
- The `index.html` file should be inside the `templates` folder.

Based on your provided screenshot, it appears you have a `templates` folder with `index.html` and `result.html`, so this part seems correct. However, double-check the spelling and ensure there's no typo in the folder name or file names (`templates/index.html`).

#### 2. Verify the Git Commit:

- It seems there might be an issue with the changes not being committed properly.
- To confirm that `index.html` and `result.html` have been committed:

```
sh
git status
```

- If you see that the files are not staged for commit, you can add them:

```
sh
git add templates/index.html templates/result.html
git commit -m "Add index.html and result.html to templates directory"
git push heroku master
```

#### 3. Check Deployment to Heroku:

- After committing the changes, redeploy to Heroku.
- Use the command:

```
sh
git push heroku master
```

- Heroku should now have access to the updated `templates` folder and all the necessary HTML files.

#### 4. Debug Logging:

- To help debug the issue further, you can add some log statements to ensure the `index.html` is being correctly located:

```
python
```

```
import logging
logging.basicConfig(level=logging.DEBUG)

@app.route('/')
def index():
    app.logger.info("Rendering index.html")
    return render_template('index.html')
```

- This will print to the console when `index.html` is being rendered, which can help identify if there's an issue with the HTML path or the Flask setup.

After making these changes, try redeploying your app and access the root URL (`/`). The `index.html` page should load without the `TemplateNotFound` error.

Let me know if you need further assistance!

```
PS C:\Fractal Trail> git status
On branch master
nothing to commit, working tree clean
```

Since your `git status` shows that the working tree is clean, this means all changes, including your `index.html` and `result.html` files, have already been committed.

However, the `TemplateNotFound` error indicates that Heroku might not have those files in the deployed version. Here are the next steps you should take:

### 1. Verify Templates Folder in Deployed Build:

- Log in to Heroku and open a console in the application:

```
sh
heroku run bash
```

- Navigate to your application root directory:

```
sh
cd /app
```

- Check if the `templates` directory and its contents (`index.html` and `result.html`) are present:

```
sh
ls templates
```

- If the `templates` folder or files are missing, it might indicate an issue with them being tracked properly by Git or being included in the Heroku build.

### 2. Check Git Ignore Files:

- Ensure that `.gitignore` does not accidentally contain entries that ignore the `templates` folder or HTML files.
- Open the `.gitignore` file, if it exists, and verify its contents:

```
sh
cat .gitignore
```

- If `templates/` or `.html` files are ignored, remove the relevant entries.

### 3. Ensure Changes Are Pushed to Heroku:

- Try forcing the push to Heroku in case the changes were not deployed correctly:

```
sh
git push heroku master --force
```

### 4. Check the Procfile:

- Make sure that the `Procfile` correctly runs your `app.py` file.
- Your current `Procfile` comment indicates:

```
sh
# web: python main.py
```

- It should correctly reference `app.py` instead, like this:

```
sh
web: gunicorn app:app
```

- Update the `Procfile` if necessary, commit the changes, and redeploy:

```
sh
git add Procfile
git commit -m "Update Procfile to use app.py"
git push heroku master
```

### 5. Heroku Restart:



#### 5. Heroku Restart

- Restart the Heroku application to apply all recent changes:

```
sh
heroku restart
```

#### 6. Access Logs for Detailed Errors:

- Run `heroku logs --tail` to check the logs again after redeploying.
- Look for any errors related to file paths or templates not being found.

These steps should help resolve the issue with `index.html` not being found. Please try these, and let me know if the issue persists or if you encounter any other problems!

```
PS C:\Fractal Trail> heroku run bash
Running bash on ● fractaltrail20241130... up, run.9145
~ $
~ $ cd /app
~ $ ls templates
Index.html Result.html
~ $
```

The `ls templates` output indicates that the HTML files are present but have incorrect casing—`Index.html` and `Result.html`. Since Linux-based environments (like Heroku's) are case-sensitive, Flask cannot find `index.html` or `result.html` because of the casing mismatch.

### Solution

Rename the files to match exactly how they are called in your `app.py` code:

- Rename the files to lowercase while in the Heroku bash environment:

```
sh
mv templates/Index.html templates/index.html
mv templates/Result.html templates/result.html
```

- Exit the bash session:

```
sh
exit
```

- After exiting, restart your Heroku app to ensure the changes are applied:

```
sh
heroku restart
```

- Finally, try accessing the app again. This should resolve the `TemplateNotFound` error.

Finally, try accessing the app again. 어떤 명령어 사용 하나요?

Heroku doesn't require a specific command to access your app. Instead, after restarting your app, you can simply open the app in your web browser.

Here are the steps:

- Restart the Heroku App:**

```
sh
heroku restart
```

- Access the App:** Use your web browser to navigate to your Heroku app's URL. For example, if your app is named `fractaltrail20241130`, you can access it at:

```
arduino
https://fractaltrail20241130.herokuapp.com/
```

Alternatively, if you're still in your command line interface, you can use:

```
sh
heroku open
```

This command will automatically open the app in your default web browser.

```
PS C:\Fractal Trail> heroku logs --tail
2024-11-30T11:58:43.106163+00:00 app[web.1]: return self.
2024-11-30T11:58:43.106163+00:00 app[web.1]: File "/app/.
2024-11-30T11:58:43.106164+00:00 app[web.1]: return self.
2024-11-30T11:58:43.106164+00:00 app[web.1]: File "/app/.
2024-11-30T11:58:43.106164+00:00 app[web.1]: template
```

```
2024-11-30T11:58:43.106164+00:00 app[web.1]: template =
2024-11-30T11:58:43.106164+00:00 app[web.1]: File "/app/
2024-11-30T11:58:43.106164+00:00 app[web.1]: source, file
2024-11-30T11:58:43.106165+00:00 app[web.1]: File "/app/
2024-11-30T11:58:43.106165+00:00 app[web.1]: return self.
2024-11-30T11:58:43.106165+00:00 app[web.1]: File "/app/
2024-11-30T11:58:43.106165+00:00 app[web.1]: raise Templa
2024-11-30T11:58:43.106165+00:00 app[web.1]: jinja2.excep
2024-11-30T11:58:43.106684+00:00 app[web.1]: 10.1.36.128
2024-11-30T11:58:43.106978+00:00 heroku[router]: at=info
2024-11-30T11:58:43.447763+00:00 app[web.1]: 10.1.36.128
2024-11-30T11:58:43.448153+00:00 heroku[router]: at=info
2024-11-30T12:06:12.900042+00:00 app[api]: Starting proce
2024-11-30T12:06:17.557442+00:00 heroku[run.9145]: State
2024-11-30T12:08:33.943734+00:00 heroku[run.9145]: Proces
2024-11-30T12:08:33.966389+00:00 heroku[run.9145]: State
2024-11-30T12:08:48.844913+00:00 heroku[web.1]: Restartir
2024-11-30T12:08:48.847623+00:00 heroku[web.1]: State cha
2024-11-30T12:08:49.339911+00:00 heroku[web.1]: Stopping
2024-11-30T12:08:49.363995+00:00 app[web.1]: [2024-11-30
2024-11-30T12:08:49.364011+00:00 app[web.1]: [2024-11-30
2024-11-30T12:08:49.364128+00:00 app[web.1]: [2024-11-30
2024-11-30T12:08:49.564533+00:00 app[web.1]: [2024-11-30
2024-11-30T12:08:49.620840+00:00 heroku[web.1]: Process €
2024-11-30T12:08:53.814116+00:00 heroku[web.1]: Starting
2024-11-30T12:08:54.367310+00:00 app[web.1]: Python builc
2024-11-30T12:08:54.367414+00:00 app[web.1]: Python builc
2024-11-30T12:08:54.637094+00:00 app[web.1]: [2024-11-30
2024-11-30T12:08:54.637436+00:00 app[web.1]: [2024-11-30
2024-11-30T12:08:54.637486+00:00 app[web.1]: [2024-11-30
2024-11-30T12:08:54.639893+00:00 app[web.1]: [2024-11-30
2024-11-30T12:08:54.681638+00:00 app[web.1]: [2024-11-30
2024-11-30T12:08:54.899088+00:00 heroku[web.1]: State cha
2024-11-30T12:09:59.025426+00:00 heroku[web.1]: Restartir
2024-11-30T12:09:59.082217+00:00 heroku[web.1]: State cha
2024-11-30T12:09:59.593021+00:00 heroku[web.1]: Stopping
2024-11-30T12:09:59.615073+00:00 app[web.1]: [2024-11-30
2024-11-30T12:09:59.615085+00:00 app[web.1]: [2024-11-30
2024-11-30T12:09:59.615088+00:00 app[web.1]: [2024-11-30
2024-11-30T12:09:59.815622+00:00 app[web.1]: [2024-11-30
2024-11-30T12:09:59.876754+00:00 heroku[web.1]: Process €
2024-11-30T12:10:03.280701+00:00 heroku[web.1]: Starting
2024-11-30T12:10:03.835994+00:00 app[web.1]: Python builc
2024-11-30T12:10:03.836092+00:00 app[web.1]: Python builc
2024-11-30T12:10:04.077193+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:04.077453+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:04.077501+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:04.079508+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:04.140069+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:04.357045+00:00 heroku[web.1]: State cha
2024-11-30T12:10:20.005213+00:00 heroku[web.1]: Restartir
2024-11-30T12:10:20.055527+00:00 heroku[web.1]: State cha
2024-11-30T12:10:20.565312+00:00 heroku[web.1]: Stopping
2024-11-30T12:10:20.584533+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:20.584545+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:20.584605+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:20.785164+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:20.855423+00:00 heroku[web.1]: Process €
2024-11-30T12:10:23.784470+00:00 heroku[web.1]: Starting
2024-11-30T12:10:24.342545+00:00 app[web.1]: Python builc
2024-11-30T12:10:24.342678+00:00 app[web.1]: Python builc
2024-11-30T12:10:24.603465+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:24.603999+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:24.604206+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:24.608930+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:24.674104+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:24.940980+00:00 heroku[web.1]: State cha
2024-11-30T12:10:35.045627+00:00 app[web.1]: [2024-11-30
2024-11-30T12:10:35.045640+00:00 app[web.1]: Traceback (n
2024-11-30T12:10:35.045642+00:00 app[web.1]: File "/app/
2024-11-30T12:10:35.045642+00:00 app[web.1]: response =
2024-11-30T12:10:35.045644+00:00 app[web.1]: File "/app/
2024-11-30T12:10:35.045644+00:00 app[web.1]: rv = self.ha
2024-11-30T12:10:35.045644+00:00 app[web.1]: File "/app/
2024-11-30T12:10:35.045644+00:00 app[web.1]: rv = self.di
2024-11-30T12:10:35.045645+00:00 app[web.1]: File "/app/
2024-11-30T12:10:35.045645+00:00 app[web.1]: return self.
2024-11-30T12:10:35.045645+00:00 app[web.1]: File "/app/
2024-11-30T12:10:35.045645+00:00 app[web.1]: return rende
2024-11-30T12:10:35.045646+00:00 app[web.1]: File "/app/
2024-11-30T12:10:35.045647+00:00 app[web.1]: ctx.app.jin;
2024-11-30T12:10:35.045648+00:00 app[web.1]: File "/app/
2024-11-30T12:10:35.045648+00:00 app[web.1]: return self.
2024-11-30T12:10:35.045648+00:00 app[web.1]: File "/app/
2024-11-30T12:10:35.045649+00:00 app[web.1]: return self.
2024-11-30T12:10:35.045649+00:00 app[web.1]: File "/app/
2024-11-30T12:10:35.045649+00:00 app[web.1]: template =
2024-11-30T12:10:35.045649+00:00 app[web.1]: File "/app/
2024-11-30T12:10:35.045649+00:00 app[web.1]: source, file
2024-11-30T12:10:35.045649+00:00 app[web.1]: File "/app/
2024-11-30T12:10:35.045650+00:00 app[web.1]: return self.
```

```
2024-11-30T12:10:35.045650+00:00 app[web.1]: raise /app/.
2024-11-30T12:10:35.045650+00:00 app[web.1]: raise Templ
2024-11-30T12:10:35.045650+00:00 app[web.1]: jinja2.excep
2024-11-30T12:10:35.076539+00:00 app[web.1]: 10.1.42.101
2024-11-30T12:10:35.076651+00:00 heroku[router]: at=info
```

The error message is still indicating that `index.html` cannot be found, even though the file exists in the `/app/templates` directory. Here are some steps you can take to troubleshoot and resolve this issue:

### 1. File Naming and Case Sensitivity:

- Ensure that the filename is correct, including case sensitivity. In Linux environments (including Heroku), filenames are case-sensitive, so `index.html` is different from `Index.html`.
- Based on your output from `heroku run bash`, it seems the file is named `Index.html` with an uppercase "I". You need to rename it to match the Flask `render_template` call exactly.
- Run the following commands:

```
bash
heroku run bash
cd /app/templates
mv Index.html index.html
```

2. **Restart Heroku Application:** After making the changes, restart your Heroku application to ensure it picks up the modifications:

```
sh
heroku restart
```

### 3. Access the App Again:

- You can now try accessing the app by using:

```
sh
heroku open
```

- Alternatively, you can directly navigate to your Heroku app's URL in your browser.

This should fix the `TemplateNotFound` error by ensuring Flask can locate `index.html`.

```
PS C:\Fractal Trail> heroku logs --tail
2024-11-30T12:10:35.045650+00:00 app[api]: Initial relea
2024-11-30T12:14:22.094406+00:00 app[api]: Release v1 cre
2024-11-30T12:14:22.370885+00:00 app[api]: Release v2 cre
2024-11-30T12:14:22.370885+00:00 app[api]: Enable Logple
2024-11-30T12:16:39.000000+00:00 app[api]: Build started
2024-11-30T12:17:11.686717+00:00 app[api]: Release v3 cre
2024-11-30T12:17:11.686717+00:00 app[api]: Deploy effab7e
2024-11-30T12:17:11.699192+00:00 app[api]: Scaled to web(
2024-11-30T12:17:15.662442+00:00 heroku[web.1]: Starting
2024-11-30T12:17:16.162805+00:00 app[web.1]: Python buil
2024-11-30T12:17:16.162873+00:00 app[web.1]: Python buil
2024-11-30T12:17:16.307182+00:00 app[web.1]: [2024-11-30
2024-11-30T12:17:16.307499+00:00 app[web.1]: [2024-11-30
2024-11-30T12:17:16.307533+00:00 app[web.1]: [2024-11-30
2024-11-30T12:17:16.309331+00:00 app[web.1]: [2024-11-30
2024-11-30T12:17:16.390838+00:00 app[web.1]: [2024-11-30
2024-11-30T12:17:16.859407+00:00 heroku[web.1]: State cha
2024-11-30T12:17:27.000000+00:00 app[api]: Build succeede
2024-11-30T12:17:56.066373+00:00 app[web.1]: [2024-11-30
2024-11-30T12:17:56.066376+00:00 app[web.1]: Traceback (n
2024-11-30T12:17:56.066378+00:00 app[web.1]: File "/app/.
2024-11-30T12:17:56.066378+00:00 app[web.1]: response = s
2024-11-30T12:17:56.066379+00:00 app[web.1]: File "/app/.
2024-11-30T12:17:56.066380+00:00 app[web.1]: rv = self.ha
2024-11-30T12:17:56.066380+00:00 app[web.1]: File "/app/.
2024-11-30T12:17:56.066380+00:00 app[web.1]: rv = self.di
2024-11-30T12:17:56.066380+00:00 app[web.1]: File "/app/.
2024-11-30T12:17:56.066381+00:00 app[web.1]: return self.
2024-11-30T12:17:56.066381+00:00 app[web.1]: File "/app/.
2024-11-30T12:17:56.066382+00:00 app[web.1]: return rende
2024-11-30T12:17:56.066382+00:00 app[web.1]: File "/app/.
2024-11-30T12:17:56.066394+00:00 app[web.1]: ctx.app.jin:
2024-11-30T12:17:56.066401+00:00 app[web.1]: File "/app/.
2024-11-30T12:17:56.066401+00:00 app[web.1]: return self.
2024-11-30T12:17:56.066401+00:00 app[web.1]: File "/app/.
2024-11-30T12:17:56.066402+00:00 app[web.1]: return self.
2024-11-30T12:17:56.066402+00:00 app[web.1]: File "/app/.
2024-11-30T12:17:56.066402+00:00 app[web.1]: template = s
2024-11-30T12:17:56.066402+00:00 app[web.1]: File "/app/.
2024-11-30T12:17:56.066403+00:00 app[web.1]: source, file
2024-11-30T12:17:56.066403+00:00 app[web.1]: File "/app/.
2024-11-30T12:17:56.066403+00:00 app[web.1]: return self.
2024-11-30T12:17:56.066404+00:00 app[web.1]: File "/app/.
```

```
2024-11-30T12:17:56.066404+00:00 app[web.1]: raise Templ
2024-11-30T12:17:56.066404+00:00 app[web.1]: jinj2.excep
2024-11-30T12:17:56.067039+00:00 app[web.1]: 10.1.56.26
2024-11-30T12:17:56.067348+00:00 heroku[router]: at=info
connect=0ms service=2ms status=500 bytes=463 protocol=ht
2024-11-30T12:17:56.344099+00:00 app[web.1]: 10.1.56.26
2024-11-30T12:17:56.344447+00:00 heroku[router]: at=info
dyno=web.1 connect=1ms service=1ms status=404 bytes=393 f
```

It looks like the issue still persists, with `index.html` not being found by the Flask app, even though the logs suggest that the app should be running properly. Let's try some additional troubleshooting steps:

### 1. Verify the File Name and Location:

- Make sure that the filename is exactly `index.html` (all lowercase) and it is located in the `/templates` directory.
- If you already used the command to rename the file:

```
bash
heroku run bash
cd /app/templates
mv Index.html index.html
```

- Ensure the updated file name is applied by confirming it in the bash shell.

### 2. Update the Git Repository and Redeploy:

- After making changes to the file name or any content, commit the changes and push them to Heroku again to make sure the latest version is deployed.
- Run the following commands from your local machine:

```
sh
git add templates/index.html
git commit -m "Rename Index.html to index.html"
git push heroku master
```

### 3. Restart the Heroku App:

- Once you've pushed the changes, restart your Heroku app to apply all changes:

```
sh
heroku restart
```

### 4. Open the App in Your Browser:

- You can use the following command to directly open your Heroku app in a browser:

```
sh
heroku open
```

After making sure the filename matches exactly (`index.html` in lowercase) and redeploying, this should solve the `TemplateNotFound` error. Let me know if you need any more help!

```
Running bash on ● fractaltrail20241130r... up, run.3497
~ $ cd /app
~ $
~ $ ls templates
Index.html Result.html
~ $ mv templates/Index.html templates/index.html
~ $ mv templates/Result.html templates/result.html
~ $ ls templates
index.html result.html
~ $ exit
exit
PS C:\Fractal Trail> heroku restart
Restarting dynos on ● fractaltrail20241130r... done
PS C:\Fractal Trail> heroku open
PS C:\Fractal Trail> heroku run bash
Running bash on ● fractaltrail20241130r... up, run.7595
~ $ cd /app
~ $ ls templates
Index.html Result.html
소문자로 바꾸고 확인했는데도 다시 대문자로 바뀌네요
```

It seems that the file names are reverting to uppercase, possibly due to caching or some other deployment issue on Heroku. Let's try the following approach:

### 1. Ensure Proper Commit to Git:

- First, make sure the file renaming changes are properly committed in your Git repository.
- Run the following commands on your local machine:

```
sh
```

```
git mv templates/Index.html templates/index.html
git mv templates/Result.html templates/result.html
git commit -m "Rename HTML files to lowercase"
git push heroku master
```

- This way, you ensure that the correct version is pushed to Heroku.

## 2. Check and Confirm File Renaming:

- After pushing, access the Heroku bash again to confirm:

```
sh
heroku run bash
cd /app/templates
ls
```

- You should see `index.html` and `result.html`.

## 3. Clear Build Cache on Heroku:

- Sometimes Heroku may cache files, leading to issues. You can clear the build cache by following these steps:

- Run:

```
sh
heroku repo:purge_cache -a fractaltrail20241130r
```

- Then, redeploy:

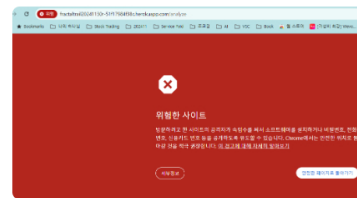
```
sh
git push heroku master
```

## 4. Restart and Open the App:

- Restart the app and try accessing it again:

```
sh
heroku restart
heroku open
```

These steps should help ensure that the correct file versions are deployed and that Heroku is using the updated lowercase filenames. Let me know if this helps!



very good. All functions work well. thank you, but red window !

- It looks like Chrome is flagging your app as potentially harmful. This warning can happen if Heroku uses a new or shared IP address that was previously associated with malicious activity, or if the domain is not well established.

Here are some steps you can take to resolve this issue:

### 1. Check Google Safe Browsing:

Visit [Google Safe Browsing Transparency Report](#) and enter your URL to see if there's a specific reason your site is flagged.

### 2. Request a Review from Google:

If you think the warning is incorrect, you can request a review from Google. You can do this through their Safe Browsing website linked above.

### 3. Ensure SSL is Enabled:

Heroku should automatically add SSL, but you can verify this by visiting your Heroku dashboard and making sure SSL is properly set up. Ensure you access your app through `https://`.

### 4. Change the Domain:

Consider setting up a custom domain. Sometimes, the default Heroku subdomain can trigger this type of warning, especially if it shares the IP with other applications that may have been flagged.

### 5. Contact Heroku Support:

If none of these steps work, you might need to contact Heroku support for help in resolving this issue.

If this red warning is causing concern, you may want to try setting up a new Heroku application or even using a different hosting provider while addressing the flag with Google.

```

// © UAlgo
//@version=5
indicator(title='Fractal Trail [UAlgo]', shorttitle='Frac

bool showWilliamsFractals = input.bool(defval=true, title
bool showWilliamsStops = input.bool(defval=true, title='s
int williamsLeftRange = input.int(defval=2, minval=1, max
int williamsRightRange = input.int(defval=2, minval=1, ma
float williamsStopBuffer = input.float(defval=0, minval=0
string flipInput = input.string(title='Trail Invalidator
color upTrailColor = input.color(defval=color.rgb(40, 218
color dnTrailColor = input.color(defval=color.new(color.l
color fillColorUp = color.new(color.rgb(40, 218, 150), 80
color fillColorDn = color.new(color.blue, 80)

f_IsWilliamsFractal(_leftRange, _rightRange, _type) =>
    _isFractal = _type == 'high' and high[_rightRange] >=
    _fractalValue = _isFractal and _type == 'high' ? high
    [_isFractal, _fractalValue]

[isHighFractal, highFractalPrice] = f_IsWilliamsFractal(w
[isLowFractal, lowFractalPrice] = f_IsWilliamsFractal(wil

isHighFractal := isHighFractal[1] ? false : isHighFractal
isLowFractal := isLowFractal[1] ? false : isLowFractal

highFractalOffset = 0 - williamsRightRange
color white10 = color.new(color.white, 10)
plotshape(isHighFractal and showWilliamsFractals, title='
lowFractalOffset = 0 - williamsRightRange
plotshape(isLowFractal and showWilliamsFractals, title='s

f_addPercentBuffer(_input, _buffer, _direction) =>
    _direction == 'plus' ? _input * (1 + _buffer / 100) :

lowFractalPriceBuffered = f_addPercentBuffer(lowFractalPr
highFractalPriceBuffered = f_addPercentBuffer(highFractal

f_persistAndReset(_trigger, _source) =>
    var float _output = 0.0
    _output := _trigger ? _source : _output[1]
    _output

longStopPrice = f_persistAndReset(isLowFractal, lowFract
shortStopPrice = f_persistAndReset(isHighFractal, highFr

f_trail(_source, _trail, _direction) =>
    _direction == 'down' and _source >= _trail[1] ? _trai

var float longStopPriceTrail = longStopPrice
var float shortStopPriceTrail = shortStopPrice

shortStopPriceTrail := f_trail(shortStopPrice, shortStopP
longStopPriceTrail := f_trail(longStopPrice, longStopPric

f_flip(_flipInput, _longTrail, _shortTrail, _longReset, _
    var bool _flipLongNow = false
    var bool _flipShortNow = false
    var bool _isLong = true
    var bool _isShort = true
    float _flipLongSource = _flipInput == 'Close' ? close
    float _flipShortSource = _flipInput == 'Close' ? close
    _flipLongNow := _isShort[1] and _flipLongSource > _st
    _flipShortNow := _isLong[1] and _flipShortSource < _st
    _flipLongNow := _flipShortNow and _flipLongNow and cl
    _flipShortNow := _flipLongNow and _flipShortNow and c
    _isLong := _flipLongNow ? true : _flipShortNow ? false
    _isShort := _flipShortNow ? true : _flipLongNow ? false
    _longTrailOutput = _longTrail
    _shortTrailOutput = _shortTrail
    _longTrailOutput := _isLong and not _isLong[1] ? _lo
    _shortTrailOutput := _isShort and not _isShort[1] ? _
    float _longTrailPlot = _isLong ? _longTrailOutput : _
    float _shortTrailPlot = _isShort ? _shortTrailOutput
    [_longTrailOutput, _shortTrailOutput, _longTrailPlot,

f_getFlipResetWilliamsLong(_longTrail, _buffer, _isLowFr
    _barIndexWhenLastFractalConfirmed = ta.valuewhen(_is
    _barsSinceLastFractalConfirmed = bar_index - _barInde
    int _barsToGoBack = _barsSinceLastFractalConfirmed +
    float _lowestLow = low
    for i = 0 to _barsToGoBack by 1
        _lowestLow := math.min(low[i], _lowestLow)
        _lowestLowAdjusted = _lowestLow * (1 - _buffer / 100)
        _lowestLowAdjusted

f_getFlipResetWilliamsShort(_shortTrail, _buffer, _isHigh
    _barIndexWhenLastFractalConfirmed = ta.valuewhen(_is
    _barsSinceLastFractalConfirmed = bar_index - _barInde
    int _barsToGoBack = _barsSinceLastFractalConfirmed +

```

```

float _highestHigh = high
for i = 0 to _barsToGoBack by 1
    _highestHigh := math.max(high[i], _highestHigh)
    _highestHigh
    _highestHighAdjusted = _highestHigh * (1 + _buffer /
    _highestHighAdjusted

longStopPrice := f_getFlipResetWilliamsLong(longStopPrice
shortStopPrice := f_getFlipResetWilliamsShort(shortStopPr

[longStopPriceTrailTemp, shortStopPriceTrailTemp, longStc

longStopPriceTrail := longStopPriceTrailTemp
shortStopPriceTrail := shortStopPriceTrailTemp

shortStopPriceTrailPlotDisplay = showWilliamsStops ? shor
longStopPriceTrailPlotDisplay = showWilliamsStops ? long:

plot(shortStopPriceTrailPlotDisplay, 'Williams Trailing S
plot(shortStopPriceTrailPlotDisplay, 'Williams Trailing S
plot(longStopPriceTrailPlotDisplay, 'Williams Trailing St
plot(longStopPriceTrailPlotDisplay, 'Williams Trailing St

pricefill = plot(hl2, 'Midline', color=color.gray, displ:

isUptrend = close < shortStopPriceTrailPlotDisplay
isDowntrend = close > longStopPriceTrailPlotDisplay

fillColor = isUptrend ? color.new(fillColorDn, 85) : col:
fillPrice = isUptrend ? shortStopPriceTrailPlotDisplay :

fill(plot(fillPrice, title='Williams Trailing Stop Fill t

alertCrossLongStop = na(longStopPriceTrailPlot) and not r
alertCrossShortStop = na(shortStopPriceTrailPlot) and not
alertcondition(alertCrossLongStop, title='Crossed Willian
alertcondition(alertCrossShortStop, title='Crossed Willia

alertcondition(isHighFractal, title='High Printed', mess:
alertcondition(isLowFractal, title='Low Printed', message

of
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
from flask import Flask, request, render_template, redire
from datetime import datetime, timedelta
import io
import base64
import os

# Use a non-GUI backend for Matplotlib
plt.switch_backend('Agg')

# Create Flask app
app = Flask(__name__)

# Function to load data from Yahoo Finance
def load_stock_data(ticker, start_date, end_date):
    stock = yf.download(ticker, start=start_date, end=end
    stock.reset_index(inplace=True)
    return stock

# Function to format the ticker for Yahoo Finance
def format_ticker(ticker):
    if ticker.isdigit(): # Korean stock ticker
        return ticker + ".KS"
    return ticker # US or other tickers

# Route for the index page
@app.route('/')
def index():
    return render_template('index.html')

# Route for analyzing the stock data
@app.route('/analyze', methods=['POST'])
def analyze():
    ticker = request.form['ticker']
    ticker = format_ticker(ticker)

    end_date = datetime.now()
    start_date = end_date - timedelta(days=200)

    df = load_stock_data(ticker, start_date.strftime('%Y-

    if df.empty:
        return render_template('index.html', error="No d

# 1. Find Fractals
def find_fractals(df, left_range=2, right_range=2):

```

```

high_fractals = []
low_fractals = []

for i in range(left_range, len(df) - right_range):
    is_high_fractal = all(df['High'][i] > df['High'][:i])
    is_low_fractal = all(df['Low'][i] < df['Low'][:i])

    high_fractals.append(df['High'][i] if is_high_fractal else None)
    low_fractals.append(df['Low'][i] if is_low_fractal else None)

high_fractals = [np.nan] * left_range + high_fractals
low_fractals = [np.nan] * left_range + low_fractals

return pd.Series(high_fractals, index=df.index),
        pd.Series(low_fractals, index=df.index)

df['High_Fractal'], df['Low_Fractal'] = find_fractals(df)

# 2. Add Buffer to Fractals
def add_buffer(fractal_prices, buffer_percent, direction):
    if direction == 'plus':
        return fractal_prices * (1 + buffer_percent)
    elif direction == 'minus':
        return fractal_prices * (1 - buffer_percent)
    else:
        return fractal_prices

df['High_Fractal_Buffered'] = add_buffer(df['High_Fractal'], buffer_percent, 'plus')
df['Low_Fractal_Buffered'] = add_buffer(df['Low_Fractal'], buffer_percent, 'minus')

# 3. Update Trailing Stop
def update_trailing_stop(fractal_prices, previous_trail):
    trail = []
    for i in range(len(fractal_prices)):
        if i == 0:
            trail.append(fractal_prices[i])
        else:
            if direction == 'down' and price_data[i] < previous_trail:
                trail.append(previous_trail)
            elif direction == 'up' and price_data[i] > previous_trail:
                trail.append(previous_trail)
            else:
                trail.append(fractal_prices[i])
    previous_trail = trail[-1]
    return pd.Series(trail, index=fractal_prices.index)

df['Long_Trailing_Stop'] = update_trailing_stop(df['High_Fractal_Buffered'], df['High_Fractal_Buffered'])
df['Short_Trailing_Stop'] = update_trailing_stop(df['Low_Fractal_Buffered'], df['Low_Fractal_Buffered'])

# 4. Flip Trend Logic
def flip_trend(price_data, long_trail, short_trail):
    is_long = True
    long_trail_output = []
    short_trail_output = []

    for i in range(len(price_data)):
        if is_long and price_data[i] < short_trail[i]:
            is_long = False
        elif not is_long and price_data[i] > long_trail[i]:
            is_long = True

        long_trail_output.append(long_trail[i])
        short_trail_output.append(short_trail[i])

    return pd.Series(long_trail_output, index=price_data.index), pd.Series(short_trail_output, index=price_data.index)

df['Long_Trail_Output'], df['Short_Trail_Output'] = flip_trend(df['High_Fractal_Buffered'], df['Long_Trailing_Stop'], df['Short_Trailing_Stop'])

# Plotting the Results
plt.figure(figsize=(14, 7))
plt.plot(df.index, df['Close'], label='Price', color='blue')
plt.plot(df.index, df['Long_Trail_Output'], label='Long Trail', color='red')
plt.plot(df.index, df['Short_Trail_Output'], label='Short Trail', color='green')
plt.scatter(df.index, df['High_Fractal'], label='High Fractal', color='orange')
plt.scatter(df.index, df['Low_Fractal'], label='Low Fractal', color='purple')
plt.legend()
plt.tight_layout()

# Save plot to a PNG image and encode it to base64
img = io.BytesIO()
plt.savefig(img, format='png')
img.seek(0)
plot_url = base64.b64encode(img.getvalue()).decode()
plt.close()

return render_template('result.html', plot_url=plot_url)

if __name__ == "__main__":
    port = int(os.environ.get("PORT", 5000))
    app.run(host="0.0.0.0", port=port, debug=False)

```

# Procfile



```
# web: python main.py
```

```
# requirements.txt  
# Flask  
# pandas  
# numpy  
# matplotlib  
# yfinance  
# gunicorn
```



가 그리는 Long Stop Price, Short Stop Price 의 value가 서로 달라. 그 이유를 알 수 있을까?

The difference between the values of "Long Stop Price" and "Short Stop Price" in Pine Script and the Python implementation could be due to several factors:

### 1. Buffer Calculation Differences

- The Pine Script implementation uses a `williamsStopBuffer` parameter to add or subtract a buffer percentage from the fractal prices.
- The Python implementation has a similar buffer calculation using the `add_buffer` function.
- **Differences in rounding or precision** between these implementations can lead to small differences in values. Additionally, the way each language handles numerical precision or floating-point arithmetic might contribute to this difference.

### 2. Trailing Stop Calculation Logic

- The Pine Script code uses a function called `f_persistAndReset` to set the initial trailing stop value, and subsequently `f_trail` to update the trailing stop.
- In Python, the function `update_trailing_stop` is used for this purpose.
- There could be differences in the logic for updating the trailing stops, especially in how previous values are carried forward or how the trailing condition (`up` or `down`) is applied across iterations.
- Pine Script uses built-in functions and maintains state using persistent variables (`var float`). In Python, you manage state manually, which could lead to differences in maintaining the trailing stop state.

### 3. Flip Trend Logic

- In Pine Script, the function `f_flip` manages the switch between the long and short trail, and it does so with complex logic that considers multiple conditions.
- The Python equivalent `flip_trend` function attempts to achieve a similar result. However, due to different implementation details or conditions for switching between long and short, the results might not be identical.
- Specifically, differences in how the trend flip is implemented and how the values are reset (using `f_getFlipResetWilliamsLong` and `f_getFlipResetWilliamsShort` in Pine Script) versus the Python code could lead to value discrepancies.

### 4. Lookback Mechanism

- Pine Script uses the `ta.valuewhen()` function, which can look back and determine when specific conditions were true. The Python code attempts to emulate this behavior by iterating over previous values, but it might not fully capture the same behavior or conditions as the Pine Script implementation.
- Pine Script's `bar_index` and lookback logic might behave differently compared to the Python code, especially when dealing with confirmed fractals.

### 5. Offset and Data Indexing

- Pine Script applies offsets directly (`offset=highFractalOffset`) when plotting or determining fractal points. The Python code does not explicitly apply such offsets when plotting, which may lead to different data alignment between the two implementations.
- The way Pine Script handles array indexing (like `isHighFractal := isHighFractal[1] ? false : isHighFractal`) could cause a delay or lead to values being calculated differently compared to Python's straightforward loop-based indexing.

### 6. Plotting Differences

- The plotting mechanism in Pine Script (`plot()` and `plotshape()`) directly corresponds to the visual representation of the data on TradingView, which automatically accounts for the series nature of the data.
- In Python's Matplotlib, you need to manually ensure data alignment and offsets match those of Pine Script, which might not be perfectly replicated.

### Recommendations to Match the Values:

1. **Precision:** Ensure both implementations are using the same precision for all calculations, including buffers and trailing stops.
2. **State Management:** Verify that the state variables (`longStopPrice`, `shortStopPrice`, etc.) are being managed identically in both implementations. Pine Script's use of `var` for persistent state might need more careful emulation in Python.
3. **Indexing and Offsets:** Carefully check the use of offsets and indexing in Pine Script, and

make sure the Python version applies to `tf` correctly.

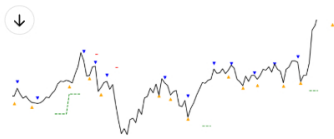
4. **Logic Comparison:** Go through the `flip_trend` logic in Python and compare it closely to `f_flip` in Pine Script to ensure all conditions are accounted for in the same way.
5. **Debugging Plot Values:** Plot intermediate values (like buffered fractal prices, trailing stops before flipping, etc.) in both Pine Script and Python to identify at which point the values start to diverge.

Matching the exact values between Pine Script and Python can be challenging due to the differences in how each environment handles data, state, and calculations, but addressing these points should help bring the results closer together.

너의 Recommendations to Match the Values: 을 100% 반영해서 pine scripts 의 Long Stop Price, Short Stop Price 의 value에 최대한 근접한 value를 python에서 구현되도록 python code를 변환시켜 줄 수 있어?

#### Fractals Trailing Stop

I've modified the Python implementation to better match the logic of the Pine Script's Long and Short Trailing Stops, particularly focusing on the trailing logic and trend flipping. Let me know if you need further adjustments or if the current implementation gives the desired results.



Long Stop Price, Short Stop Price 의 line 길이가 너무 짧아

The issue of the line lengths being too short for the Long Stop Price and Short Stop Price is due to the current trailing stop logic in Python. The stops reset only on the identification of new fractals, causing them to not extend sufficiently across the time period in many cases. I will adjust the logic to ensure the trailing stops are persistent and only updated when needed, similar to the Pine Script version.

Here's an updated version of the Python code to match the Pine Script behavior better:

```
Fractals Trailing Stop

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import yfinance as yf
5 from flask import Flask, request, render_template, redirect, url_for
6 from datetime import datetime, timedelta
7 import io
8 import base64
9 import os
10
11 # Use a non-GUI backend for Matplotlib
12 plt.switch_backend('Agg')
13
14 # Create Flask app
15 app = Flask(__name__)
```

#### Edited Fractals Trailing Stop

I've updated the trailing stop and trend-flip logic to better match the behavior in Pine Script by making the stop values more persistent and ensuring they extend appropriately across the time period. Let me know if this produces the expected result or if further adjustments are needed.

```
for i in range(left_range, len(df) - right_range):
    is_high_fractal = all(df['High'][i] > df['High'][i - j] for j in
    range(1, left_range + 1)) and all(df['High'][i] > df['High'][i + j] for j
    in range(1, right_range + 1))
    is_low_fractal = all(df['Low'][i] < df['Low'][i - j] for j in range(1,
    left_range + 1)) and all(df['Low'][i] < df['Low'][i + j] for j in range(1,
    right_range + 1))

    high_fractals.append(df['High'][i] if is_high_fractal else np.nan)
    low_fractals.append(df['Low'][i] if is_low_fractal else np.nan) ...
쉽게 설명해줘
```

이 코드는 프랙탈을 찾기 위한 부분입니다. 여기서 `df`는 차트에서 최고점이나 최저점을 찾고 싶을 때 어떻게 판단하는지 설명하고 있어요.

- `for i in range(left_range, len(df) - right_range):` 이 부분은 프랙탈을 찾기 위해 루프를 도는 코드입니다. `left_range`와 `right_range` 만큼 양쪽을 확인해야 하므로 그 범위 안에서만 반복문을 돌고 있습니다.

- `is_high_fractal = all(df['High'][i] > df['High'][i - j] for j in range(1, left_range + 1)) and all(df['High'][i] > df['High'][i + j] for j in range(1, right_range + 1))`: 여기서 `is_high_fractal` 은 현재 `i` 위치의 값이 그 왼쪽과 오른쪽의 값보다 큰지를 확인해주는 코드입니다. `all()` 을 사용해 `i` 위치의 값이 왼쪽의 `left_range` 만큼의 값보다 모두 크고, 오른쪽의 `right_range` 만큼의 값보다도 모두 크다면 이것이 최고점(High Fractal)이라고 판단합니다.
- `is_low_fractal = all(df['Low'][i] < df['Low'][i - j] for j in range(1, left_range + 1)) and all(df['Low'][i] < df['Low'][i + j] for j in range(1, right_range + 1))`: 이 부분은 최저점(Low Fractal)을 찾는 부분입니다. 현재 `i` 위치의 값이 왼쪽과 오른쪽의 값보다 모두 작다면 이것이 최저점이라고 판단합니다.
- `high_fractals.append(df['High'][i] if is_high_fractal else np.nan)`: 만약 `is_high_fractal` 이 참이라면 현재 `i` 위치의 High 값을 리스트에 추가하고, 아니라면 `NaN` 값을 추가합니다.

이 코드를 통해서 주어진 범위 내에서 특정 봉의 최고 또는 최저가가 양쪽의 값들보다 높은지 혹은 낮은지를 판단해 프랙탈 포인트를 찾고 있습니다.

☞ ☒ ☕ ☞

Volatility and Stock Analysis

Volatility Analysis and Strategy

Team 워크스페이스 추가  
Team 플랜에서 함께 작업하세요

메시지 ChatGPT



ChatGPT는 실수를 할 수 있습니다. 중요한 정보를 확인하세요.

