```
# --- Parameters & state ---
CALL_SIZE = 10
RSI_LEN = 2
state = {
    "regime": "UP",              # "UP" or "DOWN" (daily trend)
    "pos_call": 0,               # contracts
    "pos_put": 0,                # contracts
    "entry_price_call": None,
    "entry_price_put": None,
}# --- Exchange/API placeholders (implement per your broker/exchange) ---
def get_last_price(symbol: str) -> float: ...
def get_indicator(name: str, symbol: str, tf: str, **kwargs) -> float: ...
def buy_call(symbol: str, size: int): ...
def sell_call(symbol: str, size: int): ...
def buy_put(symbol: str, size: int): ...
def sell_put(symbol: str, size: int): ...
def exit_rule_hit(symbol: str) -> bool: ...   # target/trailing/RSI rollover, etc.
SYMBOL = "YOUR_SYMBOL"
```

Parameters

- CALL_SIZE: 한 번에 진입·헤지에 사용할 기본 계약 수량. 예시로 10이면 콜/풋 진입과 헤지 모두 10계약 기준으로 실행됩니다.
- RSI_LEN: RSI 지표의 기간(length). 2로 설정하면 초단기(민감) 신호를 사용합니다. 짧을수록 신호 빈도와 변동성이 높아집니다.

State dictionary

- regime: 일봉 대세 방향. "UP"(상승장)일 때 콜만 운용, "DOWN"(하락장)일 때 풋만 운용합니다. 일봉이 닫힐 때 갱신됩니다.
- pos_call: 현재 보유 중인 콜 포지션 계약 수. 0이면 콜 포지션

없음, 양수면 해당 수량만큼 보유 중입니다.

- pos_put: 현재 보유 중인 풋 포지션 계약 수. 0이면 풋 없음, 양수면 헤지 또는 주 포지션 보유 중입니다.
- entry_price_call: 콜 포지션의 진입가(평단). 손익 평가 시 기준이 되며, 포지션 청산 시 None으로 초기화합니다.
- entry_price_put: 풋 포지션의 진입가(평단). 풋 손익 평가 기준이며, 청산 시 None으로 초기화합니다.

Exchange/API placeholders

- get_last_price(symbol: str) -> float: 현재 가격(최근 체결가 또는 중간호가)을 반환합니다. 거래소/브로커 SDK에 맞춰 구현해야 합니다.
- **get_indicator(name: str, symbol: str, tf: str, kwargs) -> float: 지정 지표 값을 반환합니다.
- 예: name="RSI", tf="5m", length=RSI_LEN 또는 name="MACD_line", tf="1D", fast=1, slow=1, signal=1.
- buy_call(symbol: str, size: int): 콜 매수(진입) 주문을 실행합니다. 옵션을 쓰지 않는 환경이라면 선물/현물 기준으로 long 진입에 매핑합니다.
- sell_call(symbol: str, size: int): 콜 청산(매도) 주문을 실행합니다. 선물/현물에서는 long 포지션 청산으로 매핑합니다.
- buy_put(symbol: str, size: int): 풋 매수(진입) 주문을 실행합니다. 선물/현물 환경이라면 하방 포지션(short) 또는 델타 헤지에 해당하는 동작으로 매핑합니다.
- sell_put(symbol: str, size: int): 풋 청산(매도) 주문을 실행합니다.
- exit_rule_hit(symbol: str) -> bool: 청산 트리거를 판단하는 함수입니다.
- 예시 규칙: 고정 목표가 도달, 트레일링 스탑, 5분봉 RSI 롤오버(과매수 후 하락 전환), 모멘텀 상실 등.
- true면 해당 포지션을 청산하고 state를 초기화합니다.

Symbol and configuration

- SYMBOL = "YOUR_SYMBOL": 거래할 자산 심볼을 지정합니다.
- 예: "BTCUSDT" (바이낸스), "KRW-BTC" (업비트), 옵션 심볼이라면 거래소 표준에 맞춘 티커로 변경.
- 실전 적용 시: 심볼을 실제로 거래하는 자산명으로 바꾸고, 위 API 함수들을 해당 거래소 SDK의 메서드로 연결합니다. 주문 유형(시장가/지정가), 슬리피지 허용치, 수수료, 레버리지, 계약 승수도 설정해야 합니다.

Practical notes and common pitfalls

- 지표의 기준 가격: 코드 뼈대는 지표를 "기초자산(코인)" 가격으로 계산한다는 가정을 합니다. 옵션 프리미엄 기반으로 신호를 낼지, 기초자산으로 낼지 개발 단계에서 결정하세요.
- 손익 계산: 포인트 기준 손익을 통화 기준 손익으로 변환하려면 계약 승수(예: 100), 수수료, 슬리피지, 레버리지를 반영해야 정확합니다.
- 상태 영속화: 재시작·장애 대비를 위해 state(포지션 수량, 진입가, regime)를 파일/DB에 저장하고 복구하는 로직을 추가하세요.
- 주문 안정성: 부분체결 처리, 재시도, 오더 ID 추적, 실패·오류 로깅을 구현해야 실거래에서 안전합니다.
- 레짐 전환: 일봉 닫힘에서 regime이 바뀌면 반대 방향 포지션·헤지를 정리하여 "한 방향 프레임워크"를 유지하도록 훅(on_new_daily)을 호출하세요.
- 파라미터 변경: CALL_SIZE, RSI_LEN 등은 코드 상단에서 바꾸거나, 앱(UI) 설정으로 노출하여 클릭으로 수정 가능하게 만들 수 있습니다.

```
# --- Signals ---
def macd_centerline(symbol: str, tf: str) -> float:
    return get_indicator("MACD_line", symbol=symbol, tf=tf, fast=1, slow=1, signal=1)
def rsi2(symbol: str, tf: str) -> float:
    return get_indicator("RSI", symbol=symbol, tf=tf, length=RSI_LEN)
def signal_long(symbol: str, tf: str) -> bool:
    # Above or just crossed up over centerline
    return rsi2(symbol, tf) > macd_centerline(symbol, tf)
def signal_short(symbol: str, tf: str) -> bool:
    # Below or just crossed down under centerline
    return rsi2(symbol, tf) < macd_centerline(symbol, tf)
# --- PnL helpers (points; adapt to instrument multiplier/fees) ---
def pnl_call(current_price: float) -> float:
    if state["entry_price_call"] is None:
        return 0.0
    return current_price - state["entry_price_call"]
def pnl_put(current_price: float) -> float:
    if state["entry_price_put"] is None:
        return 0.0
```

📌 Signals 부분

이 부분은 매매 신호를 판별하는 함수들입니다.

-
- MACD 지표의 라인 값을 가져옵니다.
- 은 아주 민감한 설정으로, 사실상 단순 이동평균선처럼 작동합니다.
- 는 타임프레임(예: ,)을 의미합니다.
-
- RSI(Relative Strength Index) 지표를 길이 2로 계산합니다.
- RSI2는 초단기 과매수/과매도 신호를 빠르게 포착합니다.

•
• 조건: RSI2 값이 MACD 센터라인보다 크면 True → "롱(콜) 신호"로 판정.
• 즉, 단기 RSI가 기준선 위로 올라갔을 때 매수 신호로 해석합니다.
•
• 조건: RSI2 값이 MACD 센터라인보다 작으면 True → "숏(풋) 신호"로 판정.
• 즉, 단기 RSI가 기준선 아래로 내려갔을 때 매도 신호로 해석합니다.

📌 PnL helpers 부분
이 부분은 손익(PnL, Profit and Loss)을 계산하는 함수들입니다.
•
• 콜 포지션의 손익을 계산합니다.
• 이 None이면(즉, 아직 콜을 보유하지 않았으면) 손익은 0.0으로 반환.
• 보유 중이면 으로 손익을 계산합니다.
• 예: 진입가 100, 현재가 105 → 손익 +5 (이익).
• 진입가 100, 현재가 95 → 손익 -5 (손실).
•
• 풋 포지션의 손익을 계산합니다.
• 이 None이면(즉, 아직 풋을 보유하지 않았으면) 손익은 0.0으로 반환.
• 보유 중이면 으로 손익을 계산합니다.
• 풋은 가격이 내려갈수록 가치가 올라가기 때문에, 진입가보다 현재가격이 낮을수록 이익으로 계산됩니다.
• 예: 진입가 100, 현재가 95 → 손익 +5 (이익).
• 진입가 100, 현재가 105 → 손익 -5 (손실).

📌 질문하신 "`return 0.0 이상은 무슨 의미?""

- 여기서 ****은 포지션이 없을 때 손익을 0으로 돌려준다는 의미입니다.
- 즉, 이나 이 None이면 아직 매수/매도한 적이 없으므로 손익 계산이 불가능합니다. 그때는 그냥 손익을 0으로 처리하는 거죠.
- "이상"이라는 표현은 코드상 의미가 아니라, 질문에서 붙여주신 말로 보입니다. 코드에서는 단순히 0.0을 반환한다는 뜻입니다.

👉 정리하면:
- Signals 함수들은 RSI와 MACD를 비교해서 "롱/숏 신호"를 판별합니다.
- PnL helpers 함수들은 현재 포지션의 손익을 계산합니다. 포지션이 없으면 손익은 0으로 반환합니다.

혹시 원하시면 제가 이 부분을 실제 숫자 예시로 시뮬레이션해서 보여드릴까요? (예: 진입가 100, 현재가 95일 때 각 함수가 어떤 값을 반환하는지)

```
    # For put premium PnL you likely need option price; this placeholder uses inverse of spot movement.
    return state["entry_price_put"] - current_price
# --- Regime update on new daily candle ---
def on_new_daily():
    up = signal_long(SYMBOL, tf="1D")
    down = signal_short(SYMBOL, tf="1D")
    # Prefer explicit switch; if equal, keep prior regime
    if up and not down:
        state["regime"] = "UP"
    elif down and not up:
        state["regime"] = "DOWN"
    # Optional: when regime flips, clear hedges that violate one-direction framework
```

```
    if state["regime"] == "UP" and state["pos_put"] > 0:
        sell_put(SYMBOL, state["pos_put"])
        state["pos_put"] = 0
        state["entry_price_put"] = None
    if state["regime"] == "DOWN" and state["pos_call"] > 0:
        sell_call(SYMBOL, state["pos_call"])
        state["pos_call"] = 0
        state["entry_price_call"] = None
```

좋습니다 😎 이 코드 조각을 조목조목 설명드릴게요.

📌 Put PnL 부분

- 풋 옵션 손익(PnL) 계산을 위한 임시 코드입니다.
- 실제로는 옵션 프리미엄 가격을 사용해야 정확합니다.
- 여기서는 단순히 기초자산 가격이 내려가면 풋이 이익이라는 성질을 반영하기 위해 로 계산하는 간단한 대체 로직을 넣어둔 것입니다.
- 즉, 현재는 "풋은 가격이 내려갈수록 이익"이라는 원리를 단순화해서 표현한 placeholder입니다.

📌 Regime update on new daily candle

- 일봉이 새로 닫힐 때 대세 방향을 판정하는 함수입니다.
- → RSI2가 MACD 센터라인 위에 있으면 상승 신호.
- → RSI2가 MACD 센터라인 아래에 있으면 하락 신호.
- 조건에 따라 을 또는 으로 바꿉니다.
- 만약 두 신호가 동시에 나오거나 애매하면 기존 regime을 유지합니다.

📌 Regime 전환 시 헤지 정리

- 대세가 바뀔 때 기존 반대 포지션(헤지)을 정리하는 로직입니다.

- 상승장()으로 바뀌면, 기존에 들고 있던 풋 헤지는 무조건 청산합니다.
- 하락장()으로 바뀌면, 기존에 들고 있던 콜 포지션은 무조건 청산합니다.
- 이렇게 해야 "한 방향 프레임워크"가 유지됩니다. 즉, 상승장에서는 콜만, 하락장에서는 풋만 남도록 강제하는 원칙입니다.

☑ 정리
- 풋 손익 계산: 실제 옵션 프리미엄 대신 단순히 로 계산하는 임시 로직.
- on_new_daily 함수: 매일 일봉이 닫힐 때 RSI/MACD로 대세 방향을 판정해 regime을 또는 으로 갱신.
- 대세 전환 시 헤지 정리: 상승장에서는 풋을, 하락장에서는 콜을 무조건 청산해 한 방향 체계를 유지.

👉 즉, 이 코드는 **"일봉 대세에 따라 방향을 바꾸고, 반대 포지션은 정리해서 오직 한 방향만 유지하는 구조"**를 의미합니다.
원하시면 제가 실제 예시 시뮬레이션을 들어서 보여드릴까요? (예: 진입가 100, 현재가 95일 때 상승장에서 콜/풋이 어떻게 처리되는지)

```
# --- 5m execution loop ---
def on_new_5m():
    price = get_last_price(SYMBOL)
    if state["regime"] == "UP":
        # Entry: call-only framework
```

```
if signal_long(SYMBOL, tf="5m") and state["pos_call"] == 0:
    buy_call(SYMBOL, CALL_SIZE)
    state["pos_call"] = CALL_SIZE
    state["entry_price_call"] = price
    # If any put hedge exists, clear unconditionally
    if state["pos_put"] > 0:
        sell_put(SYMBOL, state["pos_put"])
        state["pos_put"] = 0
        state["entry_price_put"] = None
# Manage existing call
if state["pos_call"] > 0:
    if pnl_call(price) >= 0:
        # Profit or breakeven: exit on rule
        if exit_rule_hit(SYMBOL):
            sell_call(SYMBOL, state["pos_call"])
            state["pos_call"] = 0
            state["entry_price_call"] = None
            if state["pos_put"] > 0:
                sell_put(SYMBOL, state["pos_put"])
                state["pos_put"] = 0
                state["entry_price_put"] = None
    else:
        # Loss: add put hedge if not present
        if state["pos_put"] == 0:
            buy_put(SYMBOL, CALL_SIZE)
            state["pos_put"] = CALL_SIZE
            state["entry_price_put"] = price
# Hedge release on fresh call signal
if state["pos_put"] > 0 and signal_long(SYMBOL, tf="5m"):
    sell_put(SYMBOL, state["pos_put"])
    state["pos_put"] = 0
```

```
                state["entry_price_put"] = None
            # Downward manage trigger
            if signal_short(SYMBOL, tf="5m") and state["pos_call"] > 0:
                if pnl_call(price) >= 0:
                    sell_call(SYMBOL, state["pos_call"])  # lock profit
                    state["pos_call"] = 0
                    state["entry_price_call"] = None
                else:
                    if state["pos_put"] == 0:
                        buy_put(SYMBOL, CALL_SIZE)          # hedge
to stop drawdown
                        state["pos_put"] = CALL_SIZE
                        state["entry_price_put"] = price
        elif state["regime"] == "DOWN":
            # Entry: put-only framework
            if signal_short(SYMBOL, tf="5m") and state["pos_put"] == 0:
                buy_put(SYMBOL, CALL_SIZE)
                state["pos_put"] = CALL_SIZE
                state["entry_price_put"] = price
                # If any call hedge exists, clear unconditionally
                if state["pos_call"] > 0:
                    sell_call(SYMBOL, state["pos_call"])
                    state["pos_call"] = 0
                    state["entry_price_call"] = None
            # Manage existing put
            if state["pos_put"] > 0:
                if pnl_put(price) >= 0:
                    if exit_rule_hit(SYMBOL):
                        sell_put(SYMBOL, state["pos_put"])
                        state["pos_put"] = 0
                        state["entry_price_put"] = None
```

```
                    if state["pos_call"] > 0:
                        sell_call(SYMBOL, state["pos_call"])
                        state["pos_call"] = 0
                        state["entry_price_call"] = None
            else:
                # Loss: add call hedge if not present
                if state["pos_call"] == 0:
                    buy_call(SYMBOL, CALL_SIZE)
                    state["pos_call"] = CALL_SIZE
                    state["entry_price_call"] = price
        # Hedge release on fresh put signal
        if state["pos_call"] > 0 and signal_short(SYMBOL, tf="5m"):
            sell_call(SYMBOL, state["pos_call"])
            state["pos_call"] = 0
            state["entry_price_call"] = None
        # Upward manage trigger
        if signal_long(SYMBOL, tf="5m") and state["pos_put"] > 0:
            if pnl_put(price) >= 0:
                sell_put(SYMBOL, state["pos_put"])  # lock profit
                state["pos_put"] = 0
                state["entry_price_put"] = None
            else:
                if state["pos_call"] == 0:
                    buy_call(SYMBOL, CALL_SIZE)          # hedge
to stop drawdown
                    state["pos_call"] = CALL_SIZE
                    state["entry_price_call"] = price
# --- Scheduler hooks ---
# Call on_new_daily() on each new daily candle close
# Call on_new_5m() on each new 5-minute candle close (or tick
with debounce)
```

Overview of the 5-minute execution loop

이 코드는 5분봉마다 전략을 실행하며, 현재 일봉 대세(regime)에 따라 "콜 중심" 또는 "풋 중심"으로 한 방향 운용을 하고, 손실일 때만 반대 포지션으로 동일 수량 헤지합니다. 또한 새 신호가 오면 기존 헤지는 손익과 무관하게 즉시 해제하여 "한 방향 프레임워크"를 유지합니다.

Up regime logic (일봉 상승장: 콜 중심 운용)

- 가격 가져오기:
- 로 현재가를 불러옵니다.
- 진입 조건(콜만):
- 조건: 가 True이고 .
- 동작: 로 콜 10계약 진입, 과 업데이트.
- 헤지 정리: 기존 이면 으로 즉시 청산하고 , .
- 보유 콜 관리:
- 손익 ≥ 0일 때:
- 조건: .
- 동작: 가 True면 콜 청산(), 상태 초기화. 남아있는 풋 헤지도 함께 청산.
- 손익 < 0일 때:
- 조건: 손실이며 .
- 동작: 풋 10계약 헤지 진입(), 과 업데이트.
- 새 콜 신호 시 헤지 해제:
- 조건: 이고 .
- 동작: 으로 풋을 즉시 청산, , .
- 하방 관리 트리거(풋 신호 등장):
- 조건: 이고 콜을 보유 중().
- 손익 ≥ 0: 이익/본전이면 콜 청산으로 이익 고정.
- 손익 < 0: 풋 헤지가 없으면 새로 풋 10계약으로 헤지 진입.

Down regime logic (일봉 하락장: 풋 중심 운용)

- 진입 조건(풋만):

- 조건: 가 True이고 .
- 동작: 으로 풋 10계약 진입, 과 업데이트.
- 헤지 정리: 기존 이면 로 즉시 청산, , .
- 보유 풋 관리:
- 손익 $\geq$ 0일 때:
- 조건: .
- 동작: 가 True면 풋 청산, 상태 초기화. 남아있는 콜 헤지도 함께 청산.
- 손익 < 0일 때:
- 조건: 손실이며 .
- 동작: 콜 10계약 헤지 진입(), 과 업데이트.
- 새 풋 신호 시 헤지 해제:
- 조건: 이고 .
- 동작: 로 콜을 즉시 청산, , .
- 상방 관리 트리거(콜 신호 등장):
- 조건: 이고 풋을 보유 중().
- 손익 $\geq$ 0: 이익/본전이면 풋 청산으로 이익 고정.
- 손익 < 0: 콜 헤지가 없으면 새로 콜 10계약으로 헤지 진입.

Hedge release behavior

- 원칙: 새 방향(레짐에 맞는 5분 신호)이 나타나면, 반대 포지션(헤지)은 손익과 무관하게 즉시 청산합니다.
- 이유: 상승장에서는 콜 프레임워크를, 하락장에서는 풋 프레임워크를 유지해야 비용 누적과 상충 포지션을 피할 수 있습니다.

Scheduler hooks (호출 타이밍)

- 일봉 닫힘 시:
- 를 호출해 대세(regime)를 갱신하고, 필요시 반대 포지션/헤지를 정리합니다.
- 5분봉 닫힘(또는 틱 디바운스):
- 을 호출해 진입/청산/헤지/헤지해제 로직을 실행합니다.

- 의미: 전략이 정확한 캔들 클로즈를 기준으로 신호를 판정하고, 그 시점에만 행동하도록 스케줄링합니다. 틱 기반으로 쓸 경우 과도한 재진입을 막기 위해 디바운스를 권장합니다.

Practical intent and outcomes

- 한 방향 운용 유지: 일봉 상승장엔 콜만, 하락장엔 풋만을 기본으로 운용하며 손실일 때만 반대 포지션으로 동일 수량 헤지합니다.
- 신호 기반 전환: 5분봉 RSI2 vs MACD(1,1,1) 비교로 진입/관리/헤지 해제를 일관되게 수행합니다.
- 상태 관리 명확: 포지션 수량과 진입가를 로 추적하고, 청산 시 으로 초기화하여 PnL 계산과 조건 평가가 항상 일관적입니다.

원하시면 exit_rule_hit의 구체 조건(목표가, 트레일링, RSI 롤오버 등)을 선택지로 제안하고 코드에 바로 녹여드릴게요.