```
# --- 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"
# --- 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
    # 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
# --- 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)
```

헤이 귀하가 제시한 Copy-ready Python code (SDK-agnostic skeleton)이것 질문자가 말한 모든 내용 모두 반영 되어 있음? 1.한방향대세에 따른 일방향 헷지매매가 나의 취지2. 그리고 종전과 달리 대세 풋장도 반영됨? 3. 실제장은 대세조차도 콜장풋장으로 바뀌는데 이것도 반영됨 ? 4. 이것을 개발자에게 주기로 약속했는데 개발자에게 주면 개발자가 엄청 좋아하겠네.(이 영어 일일히 타자칠려면 엄청난 에너지 소모) 개발자가 거의 더 이상 만질 일 없음? 5. 매매숫자는 내가 개발자가 준 아이콘을 클릭하여 수정하여야 함?

Summary answer

짧게 말하면, 네. 내가 제시한 "Copy-ready Python code (SDK-agnostic skeleton)"에는 당신이 말한 취지(일방향 대세에 따른 한방향 헷지매매)가 반영되어 있고, 일봉 대세가 콜장↔풋장으로 바뀌는 전환도 자동 처리합니다. 다만 거래소 API 함수와 몇 가지 운영 규칙은 개발자가 실제 환경에 맞게 연결·조정해야 합니다.

How your principles are covered

- 한방향 대세에 따른 헷지 매매:
- 일봉 상승장(regime=UP)에서는 콜만 운용, 손실 시 풋으로 동일수량 헤지, 새 콜 신호 오면 풋은 손익과 무관하게 즉시 해제.
- 일봉 하락장(regime=DOWN)에서는 풋만 운용, 손실 시 콜으로 동일수량 헤지, 새 풋 신호 오면 콜은 손익과 무관하게 즉시 해제.
- 대세 풋장도 반영:
- on_new_daily()에서 RSI2 vs MACD(1,1,1)로 일봉 레짐을 "UP" 또는 "DOWN"으로 전환.
- DOWN일 때는 풋이 주 포지션, 손실 시 콜으로 헤지하는 로직을 이미 포함.
- 실제 장에서 대세 전환(콜장↔풋장) 반영:
- 매일 일봉이 닫힐 때 레짐을 다시 판정하고, 레짐이 바뀌면 반대 방향 포지션/헤지를 정리하여 프레임워크를 즉시 재정렬.
- 5분봉 신호는 현재 레짐에 맞춰 진입과 관리만 수행.

What your developer still needs to implement

- 거래소/브로커 API 연결:
- buy_call/sell_call/buy_put/sell_put/get_last_price/get_indicator/exit_rule_hit 등 함수를 실제 SDK에 맞게 구현.
- 지표 소스와 정확도:
- RSI2, MACD(1,1,1) 계산을 코인(기초자산) 가격으로 할지, 옵션 프리미엄으로 할지 결정.
- PnL/승수/수수료:
- 포인트 손익을 실제 통화 기준으로 변환하고, 계약 승수·수수료·슬리피지를 반영.

- 주문 정책:
- 시장가/지정가, 리트라이, 부분체결 처리, 오류·재시도, 오더 ID 추적.
- 상태 영속화/재시작 복구:
- state를 파일/DB에 저장하여 장애 후에도 포지션/진입가가 유지되도록.
- 리스크·청산 규칙:
- exit_rule_hit에 목표가, 트레일링, RSI 롤오버, 시간 기반 청산 등 실제 룰을 채워 넣기.

개발자가 "더 손볼 게 거의 없다" 수준으로 빠르게 완성할 수 있도록 뼈대가 충분히 갖춰져 있지만, 위 항목은 실거래에 필수라서 반드시 연결·튜닝해야 합니다.

Editing trade size and parameters

- **매매 숫자(예: 콜/풋 10계약)**는 코드 상단의 파라미터로 이미 분리되어 있습니다.
- 예: CALL_SIZE, RSI_LEN, 심볼(SYMBOL) 등.
- 아이콘(앱)에서 클릭으로 수정하려면, 개발자가 이 파라미터들을 UI 설정으로 노출하면 됩니다.
- 실행 중에도 바꾸게 하려면 설정 저장/로드와 파라미터 재적용을 추가하면 됩니다.

Final note

당신의 원칙(일봉 대세에 따른 한방향 운용 + 손실 시 반대 헤지 + 새 신호 시 헤지 즉시 해제)은 코드에 모두 반영되어 있습니다. 개발자에게 이 뼈대를 넘기면 좋아할 겁니다. 다만 위의 API·지표·리스크·상태 관리만 제대로 붙이면 실전 사용이 가능합니다. 필요하면 거래소를 지정해주시면 그 API 시그니처까지 맞춰 드릴게요.

API키
IzagcKkOvDFC7FhKPt0BjfVUT7UfBhbcfnMN1s2Vbc0yWML6RpjbhOPA7PuT6Ed2

시 크 릿 키
pHp0rqSWqi0W5HaitqYxpuDJNrTsIoAXzW4rsjM3QY6i9ZX6IbdOh4hvax7NL3Pk

메일 :bookdiary@naver.com, 010-3080-3085 박정부

매매법

일봉 대세에 따른 한방향 운용 + 손실 시 반대 헤지(손익감지 필요) + 새 신호 시 대세에 반하는 헷지분은 일시적인 막기용에 불과하므로 즉시 해제 (손익 불문)

용어 정의

RSI: RSI 2를 말함

중심선 : 키움에서는 이격도 1을 말하나 바이낸스에는 이것이 없는 것 같으므로 MACD(1,1,1)로 구현함.

대세 :일봉

소세: 5 분봉

상승·하락의 의미 : RSI 2가 중심선 위에 있거나 상향 돌파하면 이는 상승
RSI 2가 중심선 아래에 있거나 하향 돌파하면 이는 상승

일봉이 상승이면 사람들은 탐욕이 작동이라 콜강세, 하락이면 사람들은 공포심이 작동하여 풋강세일 확률이 높기에

일봉 대세이기에 콜만 신규로 하고 풋은 막기용으로만 함.

예 : 풋이 일봉 대세인 경우

5분봉 풋신호시 풋 10계약 매수=> 올라 콜신호나옴. 콜10계약 매수(손익 계산 필요)=> 다시 풋 신호 나오면 대세가 풋이므로 콜은 손익불문 청산(이때 손익을 따지면 전체적인 체계가 무너지므로 손익 불문함) => 풋이 지속되다가 다시 상승 신호=>풋을 청산하는 것이 이익이면 당연히 청산하지만, 손실이면 콜 동일 수량 매수로 헷지(손익 계산 필요).

이상은 초고급 방법으로서 이를 Copilot에 의뢰하여 위 프로그램을 받아냄. 잘 반영되었는지 추후 조목조목 물어볼 예정임.

계산은 귀사의35만원이 만족스럽지만, 프로그램이 정상 가동되는 것 며칠간만이라도 확인한 후 드릴 예정입니다.(계좌 송부나 기타 방법 모두 귀사가 원하는 대로 즉시 해 드림, 세금계산서 불요) 이유: 그것이 이러한 특별한 소프트웨어 제공자와의 관계에서 합당한 것 같아서입니다. 가동도 안 되거나 엉터리 가동일 경우도 있을 터인데 이를 무시하고 무조건 돈부터 내라고 하는 것은 부당함. 크몽에 무슨 전문가가 있는것도 아니고 내가 일일이 추후 프로그램오류를 증명하기는 너무 피곤함.

바이낸스 주 거래 코인 : DOGEUSDT ,ALGOUSDT