kucoin_interface.py

```
# =================== Imports ===================
# Library imports
import requests
import pandas as pd
import websockets
import asyncio

# Internal imports
from classes.interface_classes import Interface, DataStore, SymbolsManagerBase

# Kucoin SDK imports
from kucoin.client import WsToken
from kucoin.ws_client import KucoinWsClient


# =================== Classes ===================

class KucoinAPI(Interface):
    def __init__(self, access_key, secret_key, host="api.kucoin.com"):
        super().__init__(access_key, secret_key, host)
        self.__access_key = access_key
        self.__secret_key = secret_key
        self.__host = host

    def __get(self, path, params=None):
        return requests.get("https://{host}{path}".format(host=self.__host, path=path),
params=params).json()

    def get_symbols(self):
        return self.__get("/api/v2/symbols")

    def get_ticker(self, symbol):
        return self.__get("/api/v1/market/orderbook/level1", {"symbol": symbol})

    def get_kline_history(self, symbol, interval, limit):
        return self.__get("/api/v1/market/candles", {"symbol": symbol, "type": interval,
"limit": limit})

    def subscribe_to_candlestick(self, symbol="BTC-USDT", interval="1min",
callback_func=None, duration=6000):
        def callback(kline_data):
            print(kline_data)
            print("\n")
        if (callback_func == None):
            callback_func = callback

        async def subscribe():
            ws_token = WsToken(self.__access_key, self.__secret_key)
            ws_client = await KucoinWsClient.create(None, ws_token, callback=callback_func)
            await ws_client.subscribe('/market/candles:
{symbol}_{type}'.format(symbol=symbol, type=interval))
            await asyncio.sleep(duration)

        async_loop = asyncio.new_event_loop()
        asyncio.set_event_loop(async_loop)
        async_loop.run_until_complete(subscribe())
```

```python
    def request_trades(self, symbol="btcusdt", callback_func=None):
        pass


class KucoinSymbolsManager(SymbolsManagerBase):
    def __init__(self, interface: Interface):
        self.interface = interface

    def convert_to_dataframe(self, symbols):
        df = pd.DataFrame().from_dict(symbols)
        return df

    def filter_excluded(self, symbols: pd.DataFrame, excluded_coins: list = ["BTC-USDT",
"ETH-USDT"]):
        return super().filter_excluded(symbols, excluded_coins)

    def filter_offline(self, symbols: pd.DataFrame):
        return super().filter_offline(symbols, "enableTrading", True)
```