Projet : Framework de Backtesting d'Indices de Crypto Monnaie

Description du projet :

Conception d'un framework de backtesting pour des indices de crypto-monnaies. Ce framework vous permettra d'évaluer les performances de différents indices basés sur des thématiques particulières, en utilisant des données réelles récupérées à partir des API de CoinGecko et Binance. Le framework renverra la performance des indices construits partir d'un ensemble de paramètre sélectionné par l'utilisateur.

Consignes :

1. Récupération des données :
   - Utilisez le package 'CoinGeckoAPI' pour récupérer la liste des actifs côtés sur Binance, leur market cap, et leur tag depuis l'API CoinGecko.
   - En utilisant le package `python-binance`, récupérez les prix historiques des actifs qui composeront le backtest à partir de l'API Binance.

2. Implementation du backtest:
   - Concevez un backtester qui accepte un dictionnaire en entrée. Ce dictionnaire doit contenir :
     - Les dates de début et de fin du backtest.
     - La thématique souhaitée (définie à partir des tags).
     - La stratégie à appliquer.
     - Tout autre élément pertinent.
   - Le backtester devra renvoyer :
     - Un track record.
     - Les poids historiques de la composition de l'indice.
     - Des métriques d'analyse de performance (par exemple : le rendement total, le rendement annualisé, la volatilité, le ratio de Sharpe, etc.).

3. Structuration du code :
   - Assurez-vous d'adopter une approche orientée objet pour la structure de votre projet.
   - Divisez votre code en modules et classes distincts pour chaque fonctionnalité (par exemple : récupération de données, analyse, backtesting, etc.).
   - Commentez et documentez votre code pour faciliter sa compréhension.

Critères d'évaluation :

1. Structure du code :
   - Organisation et modularité du code.
   - Utilisation appropriée des éléments de la programmation orientée objet vus en cours.
   - Clarté et propreté du code (pas de code redondant, noms de variables explicites, etc.).

2. Fonctionnalité du framework :
   - Exactitude des données récupérées.
   - Pertinence des métriques de performance proposées.
   - Capacité du framework à gérer différentes thématiques et stratégies.

3. Documentation:
   - Présence de commentaires explicatifs.
   - Qualité et clarté de la documentation fournie (par exemple, docstrings pour les classes et méthodes).

Ressources supplémentaires :

Documentation du package `CoinGeckoAPI` : https://pypi.org/project/pycoingecko/
Documentation de l'API CoinGecko : https://pypi.org/project/pycoingecko/

Documentation du package `python-binance`: https://python-binance.readthedocs.io/en/latest/overview.html
Documentation de l'API Binance: https://binance-docs.github.io/apidocs/spot/en/#market-data-endpoints