M1 - EF - 2024-2025
Université Paris Dauphine

**Data Management - Projet 2025**
Date limite de rendu : 30/04/2025

---

Vous gérez un fond multi-asset, votre objectif est de mettre en place un pipeline de données robuste et automatisé permettant d'administrer efficacement le fonds. Le fonds doit satisfaire trois profils de clients, chacun associé à une contrainte de risque et de gestion spécifique :

- **Low risk** : les portefeuilles détenus doivent viser une volatilité annualisée d'environ 10%.
- **Low turnover** : les portefeuilles détenus ne doivent pas faire plus de 2 deals par mois.
- **High yield equity only** : maximiser le rendement en investissant exclusivement en actions (equity) sans contrainte sur la volatilité ni sur le nombre d'opérations.

Le fonds déploie **trois stratégies** distinctes pour répondre aux besoins spécifiques de ses clients. Ces stratégies pourront être calibrées sur un échantillon de données passées mais elles devront être laissées libres sur le reste de la période.

**Période d'évaluation du fond : 01-01-2023 – 31-12-2024**

Chaque lundi, le fond décide des actifs à acheter ou vendre. Un deal est caractérisé par un achat ou une vente d'un actif.

# 1 Acquisition des données

Créer un module Python autonome (data_collector.py, par exemple) qui permet de récupérer les données nécessaires au fonctionnement du fond via une (ou plusieurs) API. Le fichier doit fonctionner de manière automatique.

Le module doit également traiter les données à l'aide des méthodes vues en cours pour en faire des données exploitables :

- Traitement des valeurs manquantes (N/A)
- Traitement des valeurs aberrantes
- Transformation et mise en forme des données pour qu'elles soient conformes aux besoins de l'analyse.

Vous pouvez télécharger toutes les données dès le début de l'exécution, il n'est pas nécessaire de mettre à jour la base de données tous les jours.

## 2 Création de la structure de données du fond

Créer un fichier .py qui crée la structure de données du fond et la peuple.
Le fond doit obligatoirement disposer des tables suivantes :

- **Clients :** table contenant les informations des clients ainsi que leur profil de risque
- **Products :** table contenant les produits gérés par le fond.
- **Returns :** table contenant les returns des produits
- **Portfolios :** table contenant les portefeuilles et leur contenu
- **Managers :** table contenant les managers des portefeuilles
- **Deals :** table contenant les deals effectués par les managers en précisant quel actif est vendu et quel actif est acheté (et dans quel portefeuille)

Pour limiter le calcul, vous pouvez choisir de faire un fond avec seulement 3 clients, un par profil de risque. Le code doit cependant être modulaire pour pouvoir en ajouter d'autres si besoin.

## 3 Stratégies

Créer un fichier stratégies.py avec une fonction pour chaque profil de risque.
A vous de déterminer les inputs de vos stratégies (attention au look-forward bias) :

- Données de marché passées
- Données macros
- Performances des portefeuilles.

Dans tous les cas, chaque fonction doit retourner la décision à adopter (exemple : Lundi 22/01 : portefeuille 1 "Low risk" acheter 17, vendre 55)
La décision doit ensuite être stockée dans la table 'Deals' et les portefeuilles doivent être mis à jour en conséquence.
Vous pouvez créer un fichier Base_update.py qui permet de stocker les résultats dans la base afin de rendre le code plus clair.

## 4 Performance

Créer un fichier .py qui affiche les métriques de performance pertinentes pour évaluer le fond.
Ce module doit être lancé à la fin de la période et doit interroger directement la base.
A vous de déterminer les informations les plus pertinentes à retourner(ratio de Sharpe par portefeuille, beta par rapport à un benchmark, manager le plus efficace, graphe des différents portefeuilles ...)

## 5 Fichier Main

L'intégralité du projet doit s'exécuter depuis un notebook jupyter.
Le fichier main doit lancer :

- Le module de création et de peuplement de la base.
- Le module de collecte et de traitement des datas
- Pour tous les jours de la période, si c'est un lundi, lancer les modules de stratégie et de mise à jour de la base.
- A la fin de l'exécution lancer le module permettant d'afficher les métriques de performance.

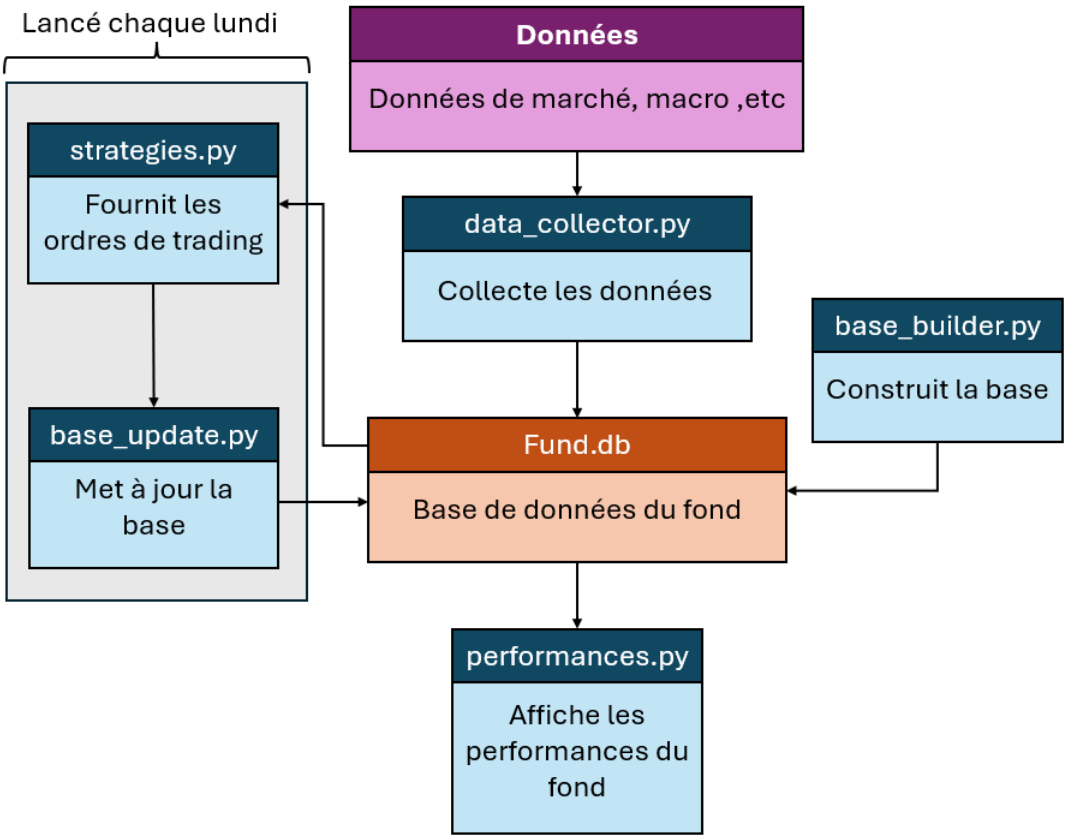


**Schéma de la structure du projet**

## Rendu final

Le rendu final à envoyer dans un .zip par mail doit inclure :

1. Le module d'acquisition et de préparation des données
2. Le module de création/peuplement de la base
3. Le fichier stratégies.py contenant les stratégies
4. Le fichier d'évaluation de performance
5. Le fichier Main.ipynb
6. Un document pdf (maximum 10 pages) contenant l'explication de votre code, des données utilisées ainsi que des stratégies
7. Un fichier requirement.txt avec les packages nécessaires au projet

L'objectif ici n'est pas d'évaluer la performance des stratégies de trading, mais de construire un pipeline de données clair et efficace, vous serez donc essentiellement notés sur les choix de structure de données et sur la façon dont votre code est agencé.
Pensez à inclure des commentaires et des informations claires sur le fonctionnement de votre code. L'utilisation de la programmation orientée objet sera valorisée si l'emploi est pertinent.

Libre à vous d'ajouter de nouveaux profils de clients, de raffiner la base de données, d'optimiser la structure du projet. Toute initiative pertinente sera pris en compte dans la notation sous forme de bonus.