Financial Model Module Documentation

1 Overview

This module defines the structure and implementations for financial modeling in a risk assessment context. It includes abstract base classes for financial data providers and financial models, as well as concrete implementations.

2 Classes

2.1 FinancialDataProvider (ABC)

An abstract base class that defines the interface for providing financial data about assets.

2.1.1 Methods

- `get_asset_value(self, asset: Asset, currency: str) -> float`
  - Returns the current value of the asset in the specified currency.
  - Parameters:
    - `asset`: The asset to value.
    - `currency`: The currency in which to express the value.
- `get_asset_aggregate_cashflows(self, asset: Asset, start: datetime, end: datetime, currency: str) -> float`
  - Returns the expected sum of cashflows generated by the asset between the start and end dates, in the specified currency.
  - Parameters:
    - `asset`: The asset to calculate cashflows for.
    - `start`: The start date for the cashflow calculation.
    - `end`: The end date for the cashflow calculation.
    - `currency`: The currency in which to express the cashflows.

2.2 FinancialModelBase (ABC)

An abstract base class that defines the interface for financial models used in risk assessment.

2.2.1 Methods

- `damage_to_loss(self, asset: Asset, impact: np.ndarray, currency: str)`
  - Converts the fractional damage of the specified asset to a financial loss.
  - Parameters:
    - `asset`: The affected asset.
    - `impact`: An array of fractional damage values.
    - `currency`: The currency in which to express the loss.
- `disruption_to_loss(self, asset: Asset, impact: np.ndarray, year: int, currency: str)`
  - Converts the fractional annual disruption of the specified asset to a financial loss.
  - Parameters:
    - `asset`: The affected asset.
    - `impact`: An array of fractional disruption values.
    - `year`: The year for which to calculate the loss.
    - `currency`: The currency in which to express the loss.

2.3 FinancialModel

A concrete implementation of `FinancialModelBase` that uses a `FinancialDataProvider` as its source of information.

2.3.1 Attributes

- `data_provider`: An instance of `FinancialDataProvider`

2.3.2 Methods

- `__init__(self, data_provider: FinancialDataProvider)`
  - Initializes the FinancialModel with a data provider.
- `damage_to_loss(self, asset: Asset, impact: np.ndarray, currency: str)`
  - Implements the abstract method from `FinancialModelBase`.
  - Calculates loss by multiplying the asset value by the impact.
- `disruption_to_loss(self, asset: Asset, impact: np.ndarray, year: int, currency: str)`
  - Implements the abstract method from `FinancialModelBase`.
  - Calculates loss by multiplying the asset's annual cashflows by the impact.

2.4 CompositeFinancialModel

A concrete implementation of `FinancialModelBase` that allows for different financial models to be used based on asset type.

2.4.1 Attributes

- `financial_models`: A dictionary mapping asset types to `FinancialModelBase` instances.

2.4.2 Methods

- `__init__(self, financial_models: Dict[type, FinancialModelBase])`
  - Initializes the CompositeFinancialModel with a dictionary of financial models.
- `damage_to_loss(self, asset: Asset, impact: np.ndarray, currency: str)`
  - Delegates the calculation to the appropriate financial model based on the asset type.
- `disruption_to_loss(self, asset: Asset, impact: np.ndarray, year: int, currency: str)`
  - Delegates the calculation to the appropriate financial model based on the asset type.

3 Usage

1. Implement a concrete `FinancialDataProvider` to provide actual financial data for assets.
2. Create instances of `FinancialModel` or `CompositeFinancialModel` as needed.
3. Use these models in risk assessment calculations to convert physical impacts to financial losses.

4 Example

```
class MyDataProvider(FinancialDataProvider):
    def get_asset_value(self, asset, currency):
        # Implementation
        pass

    def get_asset_aggregate_cashflows(self, asset, start, end,
    currency):
```

```
        # Implementation
        pass

data_provider = MyDataProvider()
financial_model = FinancialModel(data_provider)

# Use in risk assessment
asset = Asset(...)
impact = np.array([0.1, 0.2, 0.3])  # Example impact values
loss = financial_model.damage_to_loss(asset, impact, "USD")
```

5 Notes

- This module provides a flexible structure for financial modeling in risk assessment.
- The `CompositeFinancialModel` allows for tailored financial models for different asset types.
- Implementations should handle currency conversions if necessary.
- Error handling and input validation should be considered in concrete implementations.