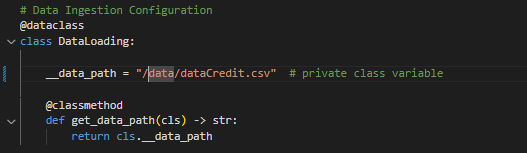
فراخوانی داده

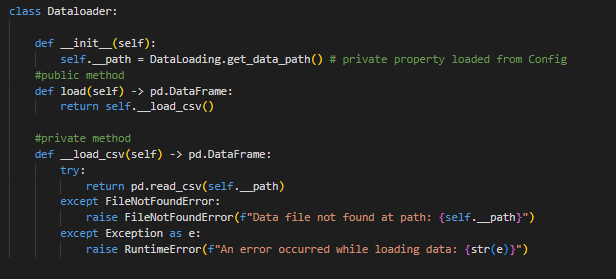


1. Read Path from config.py( Define data loading Class)

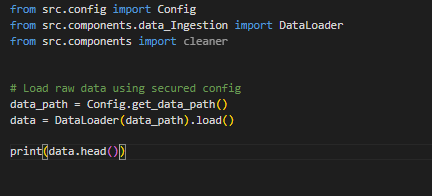


1. Read Path from config.py by data\_ingestion module(define Dataloader class)

* The data path encapsulated inside the Config module.
* The data\_ingestion /DataLoader class is fully responsible for accessing and validating the path.
* No path hardcoded in main.py.



1. Call from main.py



**✅ Summary of Benefits**

* ✔️ The file path is now hidden from the entry point.
* ✔️ Easy to update the path centrally in config.py.
* ✔️ Safer and cleaner for production.

**📘 Design Justification for load() and \_\_load\_csv() Separation**

The DataLoader class separates the load() method (public) and the \_\_load\_csv() method (private) to follow best practices in object-oriented design, particularly regarding encapsulation and modularity.

1. **Separation of Interface and Implementation**  
   The load() method provides a clean public interface for users of the class, while \_\_load\_csv() contains the internal logic for reading data from a CSV file. This makes it easier to change the underlying implementation in the future without affecting the external usage.
2. **Scalability and Extensibility**  
   The design allows for future support of multiple file formats (e.g., JSON, Excel) by simply adding private methods like \_\_load\_json() or \_\_load\_excel() and dispatching them from the load() method based on the file extension.
3. **Encapsulation**  
   Marking \_\_load\_csv() as a private method ensures that the internal logic remains protected and cannot be accessed or misused directly from outside the class.
4. **Improved Maintainability and Testing**  
   Isolating the file-loading logic in a dedicated method improves code readability and simplifies unit testing for internal behaviors.

This design adheres to the *Single Responsibility Principle* and promotes clean, modular code that is easier to maintain, scale, and test over time.

