Modern Python Recreation of COBOL Overnight Batch Interest Processing

Author: Frostbyte Research | Date: November 25, 2025

Traditional banking relies on COBOL batch jobs (e.g., CALC-INT-OVERNIGHT) for daily interest calculations. This white paper details a Python implementation that modernizes these critical processes while preserving precision, rounding rules, and audit requirements.

Introduction

Batch processing is the heartbeat of banking. Every night, millions of accounts must be processed to apply interest, clear transactions, and generate reports. In the mainframe world, this is handled by JCL and COBOL. This paper demonstrates how to achieve the same reliability and throughput using Python.

Architecture

Core Components

- Batch Processor (batch/interest_batch.py): The engine that iterates through all accounts, applying the interest calculation logic.
- Interest Calculator: A specialized module that implements the "Round Half Up" rule mandated by banking regulations, using Python's decimal context.
- CSV Data Store: Simulates VSAM datasets, providing a flat-file persistence layer that is easy to audit and backup.

Process Flow

1. Load: Reads account data from the master CSV file.
2. Process: For each account, calculates daily interest based on the principal and rate.
3. Update: Modifies the account balance in memory.

4. Commit: Writes the updated records back to the CSV file in a safe, atomic operation.

Key Features

- Regulatory Compliance: Strictly adheres to financial rounding rules (ROUND_HALF_UP) to ensure interest payments are exact to the penny.
- Performance: Optimized for sequential processing, capable of handling large datasets efficiently.
- Auditability: Generates detailed logs of every interest application, including the before and after balances.
- Resilience: Includes error handling to skip malformed records without crashing the entire batch job.

COBOL Emulation Details

- Rounding: Replicates the behavior of COBOL's COMPUTE ROUNDED clause.
- Sequential Access: Mimics the READ NEXT RECORD pattern of VSAM files.
- Reporting: Produces summary statistics (total records processed, total interest paid) similar to JCL job outputs.

Conclusion

This batch processing implementation demonstrates that Python is more than capable of handling the heavy lifting of back-office banking. By combining modern language features with disciplined financial engineering, we can replicate the reliability of the mainframe at a fraction of the cost.