I'll help you create Python code to merge STDF files while keeping only the final test results for each part. Let me create a comprehensive solution using the Semi-ATE STDF package.

I've created a comprehensive Python script that merges STDF files while removing redundant test results. Here's what the code does:

**Key Features:**

1. **Intelligent Merging**: Processes files in chronological order and keeps only the final test result for each part/test combination
2. **Part Identification**: Uses HEAD\_NUM, SITE\_NUM, and PART\_NUM to uniquely identify parts
3. **Test Identification**: Uses TEST\_NUM and TEST\_TXT to uniquely identify tests
4. **Record Type Handling**: Properly handles different STDF record types (PIR, PRR, PTR, MPR, FTR, etc.)
5. **Latest Results Priority**: When the same part/test combination appears in multiple files, the version from the latest file is kept

**Usage:**

# Basic usage

python stdf\_merger.py file1.stdf file2.stdf file3.stdf -o merged.stdf

# With wildcards

python stdf\_merger.py \*.stdf -o final\_results.stdf

# Verbose output

python stdf\_merger.py file1.stdf file2.stdf -o merged.stdf -v

**How it works:**

1. **File Processing**: Reads each STDF file and categorizes records by type
2. **Duplicate Detection**: Identifies duplicate part/test combinations across files
3. **Latest Selection**: For duplicates, selects the record from the chronologically latest file
4. **Proper Ordering**: Maintains proper STDF record ordering (PIR → Tests → PRR for each part)
5. **File Generation**: Creates a new STDF file with deduplicated results

**Installation Requirements:**

First install the Semi-ATE package:

pip install Semi-ATE

The script handles your exact use case where:

* File 1 is the largest (complete test)
* File 2 is smaller (retests of failed parts/tests)
* File 3 is even smaller (further retests)

The final merged file will contain only the latest test result for each part, effectively giving you the final test status after all retesting iterations.