This program works the same as **myproject**, but it only operates on the first three domains in TPTP. This limitation is there just to confirm that everything runs correctly before expanding to a larger dataset.

The **tptp_test_iterative.py** script processes logical formulas from the TPTP library, extracting predicates, constants, variables, and functions using regex patterns. It takes an iterative approach to classify function parameters, avoiding the recursion depth issues that could arise with deeply nested structures. The extracted information is stored in JSON format, making it easy to analyze later.

The **extract_data.py** script is designed to pull out key components from JSON files containing logical formulas. It identifies predicates, constants, and variables by applying regex-based extraction methods. Unlike other scripts that might handle formulas at an individual level, this one aggregates the extracted data across an entire domain before saving it, giving a more structured overview of the dataset.

The **extract_data_with_id.py** script works similarly to **extract_data.py** but retains a more detailed structure by keeping the extracted elements organized by page identifier. Instead of merging everything at the domain level, it preserves which formulas belong to which specific part of the dataset, making it easier to track sources and maintain clarity when analyzing the extracted information.

The **parsed_tptp_iterativ.py** script performs formula extraction like **tptp_test_iterative.py** but ensures a structured parsing process. It follows an iterative approach for classifying function parameters while efficiently handling equations and logical structures that may not always contain predicates. This ensures consistency in parsing, even when formulas present complex structures.

The **run_spiders.py** script automates the execution of multiple Scrapy spiders sequentially. Instead of running all spiders at once, it ensures that one spider completes before the next one starts. This is done using **Scrapy's CrawlerProcess** along with an asyncio reactor, allowing for structured and controlled execution of web scraping tasks without overwhelming system resources.

The **tptp_test.py** script is another version of **tptp_test_iterative.py**, but it takes a recursive approach instead of an iterative one when classifying function parameters. Like the iterative version, it processes formulas to extract predicates and equations while storing the parsed data in JSON format. However, because it uses recursion, it may face performance limitations when handling deeply nested function structures.