

# Code Operations

Code operations in DocETL allow you to define transformations using Python code rather than LLM prompts. This is useful when you need deterministic processing, complex calculations, or want to leverage existing Python libraries.

## Motivation

While LLM-powered operations are powerful for natural language tasks, sometimes you need operations that are:



- Deterministic and reproducible
- Integrated with external Python libraries
- Focused on structured data transformations
- Math-based or computationally intensive (something an LLM is not good at)

Code operations provide a way to handle these cases efficiently without LLM overhead.

## Types of Code Operations

### Code Map Operation

The Code Map operation applies a Python function to each item in your input data independently.

 **Example Code Map Operation** 

```
- name: extract_keywords
  type: code_map
  code: |
    def transform(doc) -> dict:
      # Your transformation code here
      keywords = doc['text'].lower().split()
      return {
        'keywords': keywords,
        'keyword_count': len(keywords)
      }
```

The code must define a `transform` function that takes a single document as input and returns a dictionary of transformed values.

## Code Reduce Operation

The Code Reduce operation aggregates multiple items into a single result using a Python function.



### Example Code Reduce Operation



```
- name: aggregate_stats
  type: code_reduce
  reduce_key: category
  code: |
    def transform(items) -> dict:
        total = sum(item['value'] for item in items)
        avg = total / len(items)
        return {
            'total': total,
            'average': avg,
            'count': len(items)
        }
```

The transform function for reduce operations takes a list of items as input and returns a single aggregated result.

## Code Filter Operation

The Code Filter operation allows you to filter items based on custom Python logic.



### Example Code Filter Operation



```
- name: filter_valid_entries
  type: code_filter
  code: |
    def transform(doc) -> bool:
        # Return True to keep the document, False to filter it out
        return doc['score'] >= 0.5 and len(doc['text']) > 100
```

The transform function should return True for items to keep and False for items to filter out.

## Configuration

## Required Parameters

- **type:** Must be "code\_map", "code\_reduce", or "code\_filter"
- **code:** Python code containing the transform function. For map, the function must take a single document as input and return a document (a dictionary). For reduce, the function must take a list of documents as input and return a single aggregated document (a dictionary). For filter, the function must take a single document as input and return a boolean value indicating whether to keep the document.

## Optional Parameters

| Parameter               | Description  | Default  |
|-------------------------|--|--|
| drop_keys               | List of keys to remove from output (code_map only)                       | None   |
| reduce_key              | Key(s) to group by (code_reduce only)                                    | "_all"   |
| pass_through            | Pass through unmodified keys from first item in group (code_reduce only) | false  |
| concurrent_thread_count | The number of threads to start   | the number of logical CPU cores (os.cpu_count()) |