# Geist

## None

*None*

*None*

# Table of contents

1. Welcome to Geist documentation 3
   1.1 Features 3
   1.2 Demo for SciPy 2024 3
2. Setup 4
3. Geist Templates 5
   3.1 Geist Templates 5
   3.2 Tags 0
4. CLI 0
   4.1 Command report 0
   4.2 Command create 0
   4.3 Command destroy 0
   4.4 Command export 0
   4.5 Command graph 0
   4.6 Command load 0
   4.7 Command query 0
5. Python API 0
   5.1 Class Connection 0
   5.2 Functions 0
6. Changelog 0
   6.1 v0.3.1 0
   6.2 v0.3.0 0
   6.3 v0.2.1 0
   6.4 v0.2.0 0
   6.5 v0.1.0 0
   6.6 v0.0.1 0

# 1. Welcome to Geist documentation

Geist is a new templating language for declarative data manipulation, query, and report generation. Building on the Jinja2 template engine, Geist is designed to support diverse data backends and query engines via predefined tags and filters, and may be extended with custom tags. A single Geist template may include multiple queries expressed in different languages, e.g. SQL and SPARQL, to leverage the strengths of each for clarity and ease of maintenance. Because Geist both can generate reports in diverse formats and perform inserts and updates on new or existing databases during template expansion, Geist templates may orchestrate data extraction, transformation, and load operations spanning multiple tools and data storage systems. The Geist Python package can be installed easily and accessed via the command line. If your dataset is stored in DuckDB and SPARQL queries are more suitable for your problem, then Geist might be for you! Check out our Poster for SciPy 2024!

At the moment, Geist supports DuckDB and RDFLib. More types of data backends will be available in the near future.

## 1.1 Features

Both CLI and Python API provide the following features:

- report feature: expand a report using dataset(s)
- create feature: create a new dataset
- destroy feature: delete a dataset
- export feature: export a graph
- graph feature: visualize a dataset
- load feature: import data into a dataset
- query feature: perform a query on a dataset

## 1.2 Demo for SciPy 2024

**A Geist report that employs two different query languages**. We demonstrate how Geist can be used to extract triples from a relational database, store them as a RDF dataset, and perform SPARQL queries on it. Instead of purely in-memory operations, Geist can be used to migrate data. With the hamming numbers dataset stored in DuckDB as an input, we generate a report to describe the original dataset and the subgraph extracted via SQL and SPARQL queries using a single Geist script.

# 2. Setup

Before installing Geist, please make sure Graphviz is installed.

**Example: GitHub Codespaces**

```
sudo apt-get update && sudo apt-get install -y graphviz graphviz-dev
```

**Example: Google Colab (Jupyter Notebook)**

```
apt install libgraphviz-dev
```

Install Geist:

```
pip install geist-p
```

To check Geist is working, run `geist` in the command line. You should get the following output:

```
Usage: geist [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  create   Create a new dataset
  destroy  Delete a dataset
  export   Export a dataset
  graph    Visualize a dataset
  load     Import data into a dataset
  query    Perform a query on a dataset
  report   Expand a report using dataset(s)
```

# 3. Geist Templates

## 3.1 Geist Templates

### 3.1.1 What is a Geist template?

A Geist template is a text file without a specific extension requirement although adding a `.geist` extension is recommended. It is an extension of a Jinja template, therefore it follows the default Jinja delimiters:

- `{% ... %}` for Statements
- `{{ ... }}` for Expressions to print to the template output
- `{# ... #}` for Comments not included in the template output

### 3.1.2 How to write a Geist template?

A Geist template relies on tags and filters.

#### Tags

Tags are used within the statements, i.e., `{% ... %}`. There are two types of tags, `StandaloneTag` and `ContainerTag`. While the `StandaloneTag` does not require a closing tag, the `ContainerTag` does. Besides the Jinja predefined tags (e.g., `for`), Geist supports the following tags:

`StandaloneTag`:

- destroy
- graph
- graph2
- use

`ContainerTag`:

- create
- load
- query
- component
- map
- html
- img
- table

Custom tags can be defined through files with the use tag.

#### Filters

Filters are used to modify variables. Each filter can only take one variable as input. Multiple filters can be applied to a single variable in sequence. For example, `{{ var|filter1|filter2|filter3 }}` denotes the variable `var` will be processed through `filter1` first, then `filter2`, and `filter3` at the end.

Besides the Jinja predefined filters, Geist supports the following filters:

- **head**: extract the first 5 rows of a Pandas data frame
- **csv2df**: convert a CSV string to a Pandas data frame
- **dict2df**: convert a dictionary to a Pandas data frame
- **json2df**: convert a JSON string to a Pandas data frame
- **json2dict**: convert a JSON string to a dictionary
- **df2json**: convert a Pandas data frame to a JSON string
- **df2htmltable**: convert a Pandas data frame to an HTML table
- **escape_quotes**: escape both double and single quotation marks
- **process_str_for_html**: preprocess a string to be displayed within an HTML document, e.g., replace `<` with `&lt`

## 3.1.3 How to execute (expand) a Geist template?

CLI Python API

*report* command can expand a report (Geist template) using dataset(s).

Here are options of the *report* command:

```
Usage: geist report [OPTIONS]

Expand a report using dataset(s)

Options:
-ifile, --inputfile FILENAME   Path of the file containing the report
                                template to expand  [required]
-oroot, --outputroot TEXT      Path of the directory to store the expanded
                                report (default: current directory)
-so, --suppressoutput BOOLEAN  Suppress output or not (default: False)
-a, --args <TEXT TEXT>...      Arguments to be passed to the report
                                 template, e.g., (arg, value) indicates that
                                 {{ arg }} in the report template will be
                                 replaced by value
--help                         Show this message and exit.
```

**Example 1: expand a report from stdin**

```
geist report << END_TEMPLATE

{% create "test", datastore="rdflib", inputformat="nt", isfilepath=False %}
    <http://example.com/drewp> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
    <http://example.com/drewp> <http://example.com/says> "Hello World" .
{% endcreate %}

{% query "test", datastore="rdflib", isfilepath=False as all_triples %}
    SELECT ?s ?p ?o
    WHERE {
        ?s ?p ?o
    }
    ORDER BY ?s ?p ?o
{% endquery %}

{% for _, row in all_triples.iterrows() %}
    Subject: {{ row["s"] }}, Predicate: {{ row["p"] }}, Object: {{ row["o"] }}.
{% endfor %}

{% destroy "test", datastore="rdflib" %}

END_TEMPLATE
```

**Example 2: expand a report from a file**

```
geist report --inputfile report.geist
```

Here is the report.geist file:

```
{% create "test", datastore="rdflib", inputformat="nt", isfilepath=False %}
    <http://example.com/drewp> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
    <http://example.com/drewp> <http://example.com/says> "Hello World" .
{% endcreate %}

{% query "test", datastore="rdflib", isfilepath=False as all_triples %}
    SELECT ?s ?p ?o
    WHERE {
        ?s ?p ?o
    }
    ORDER BY ?s ?p ?o
{% endquery %}

{% for _, row in all_triples.iterrows() %}
    Subject: {{ row["s"] }}, Predicate: {{ row["p"] }}, Object: {{ row["o"] }}.
{% endfor %}

{% destroy "test", datastore="rdflib" %}
```

**Example 3: expand a report from a file with external arguments**

```
geist report --inputfile report.geist --args sentence "Hello World" --args feeling Happy
```

Here is the report.geist file:

```
{% create "test", datastore="rdflib", inputformat="nt", isfilepath=False %}
    <http://example.com/drewp> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
    <http://example.com/drewp> <http://example.com/says> "{{ sentence }}" .
    <http://example.com/drewp> <http://example.com/feels> "{{ feeling }}" .
{% endcreate %}

{% query "test", datastore="rdflib", isfilepath=False as all_triples %}
    SELECT ?s ?p ?o
    WHERE {
        ?s ?p ?o
    }
```