Python

Description

The **Python** component creates a new dataset with a specified structure and populates it with data by running Python code. The code can use data from the input ports to populate the output dataset.

You can define the structure of the output dataset in the node's configuration wizard or dynamically in the Python code.

> **Note:** To use **Python** nodes, you might need to configure Megaladata and install Python first. For more information about using Python in Megaladata, see Features and Limitations.

Ports

Input

- **Input data source**: A data table. This input is optional.
- **Input variables**: A set of variables. This input is optional.
- **Add another port**: Creates an additional input port for a data table, **Input data source N**, where N is the port's sequential number.

Output

- **Output data set**: A data table.

Configuration wizard

The configuration wizard has the following pages:

- **Configure input columns** (for each data table port that has no input data)
- **Configure Python output table columns**
- **Python**: Enter the code and preview the results

Configure Python output table columns

On this page, you can manually define the columns for the output dataset. Alternatively, you can define them dynamically by running Python code.

The page has the following commands:

- **Allow creating output columns in script**: Select this checkbox to allow dynamic creation, modification, and deletion of output columns through your script.
- **Add**: Adds a new column with default parameters.
- **Clone**: Adds a new column with the same parameters as the selected column.
- **Edit** (F2): Opens a dialog to edit the parameters of the selected column.
- **Move up** (Ctrl + Up): Moves the selected column one position up.
- **Move down** (Ctrl + Down): Moves the selected column one position down.
- **Filter**: Allows you to filter the list of columns.
- **Delete all** (Shift + Delete): Deletes all columns.
- **Delete** (Delete): Deletes the selected column.

Python

The **Python** page contains the code editor for the node.

- To run the script in a separate process (the *Python interpreter process*), select the **Start in separate process** checkbox. Otherwise, the script runs inside the Megaladata process.

> **Note:** When a Python script runs inside the Megaladata process, only one **Python** node can run at a time. The next **Python** node in the queue waits for the current one to finish. The maximum waiting time is set by the **Start wait timeout (ms)** parameter. By default, there is no timeout. If the timeout is exceeded, the node stops running and returns an error. For more information see Searching for Python modules inside the Megaladata process.

- Click **Preview** to display the first 100 rows of the resulting dataset and the output console.

> **Note:** When you click **Preview**, all input ports are activated. The **Preview** window opens after they successfully activate.

Accessing port data and other built-in objects from Python code

The following objects are avaialable in the code execution context to let you access port data and other built-in objects:

- Input datasets (`InputTables` , `InputTable`)
- Input variables (`InputVariables`)
- Output dataset (`OutputTable`)
- Enumerations (`DataType` , `DataKind` , `UsageType`)

These objects are imported from the built-in `builtin_data` module. By default, an `import` statement for these objects is added to the code.

Error handling

If there are syntax or runtime errors, a message appears that indicates the location of the error in the code when you open the **Preview** window or run the node.

Global functions

The following global functions are available in the node's execution context. No additional `import` statements are required to use them.

- `filestoragepath` : Takes a path in the file storage and converts it to an absolute path in the local file system.

▼ **Example:**

```
from builtin_data import OutputTable
import json

with open(filestoragepath("data_file.json")) as f:
    data = json.load(f)
OutputTable.Append()
OutputTable.Set(0, data)
```

- `getLocale` : Returns the locale name of the node where the function is executed, in ICU format. This function takes no arguments.

Read on: Python: Input Datasets

Articles in Section:

- Input Datasets
- Input Variables
- Output Datasets

- Enumerations
- Output Console
- Full API Description
- Hotkeys
- Restrictions