Accessing Output Datasets

To access the data in Python's output ports, use dedicated objects: `OutputTable` and `OutputTables[]`:

- Use the `N` index of a corresponding **Output port N** to distinguish data sources: `OutputTables[N]`
- The first output data port, present by default, has index `0`, not displayed in the port's name. (I.e., the port name shown is just **Output dataset**.) To access data in this default port, use a special object: `OutputTable`

OutputTable properties

▼ Columns

`Columns`

Contains the read-only iterable collection of columns, implements the `Mapping` and `Sequence` protocols, and returns a value of type `ColumnsClass`. Elements can be accessed via bracket notation `[]` by names and by indexes. If the **Allow creating output columns in script** checkbox is marked, the collection elements are of type `ConfigurableOutputColumnClass`; otherwise, they are `OutputColumnClass`. Both of these types are inherited from `ColumnClass` and implement the `Sequence` protocol (see Full API Description).

▼ ColumnCount

`ColumnCount`

Contains the read-only count of columns in the output dataset. Returns a value of type `int`.

▼ RowCount

`RowCount`

Contains the read-only count of rows in the output dataset. Returns a value of type `int`.

OutputTable methods

▼ Get

`Get(row, col)`

- `row` : The row index. Accepts a value of type `int` .
- `col` : The column index or name. Accepts values of type `int` or `str` .

The method returns the value of the specified column in the specified row. The returned value can be of the following types: `bool` , `int` , `float` , `str` , `datetime.datetime` , or `None` .

▼ IsNull

`IsNull(row, col)`

- `row` : The row index. Accepts a value of type `int` .
- `col` : The column index or name. Accepts values of type `int` or `str` .

The method returns a boolean value `true` if the column in the specified row has a missing value. Otherwise, it returns `false` .

▼ GetColumn

`GetColumn(col)`

- `col` : The column index or name. Accepts values of type `int` or `str` . If the **Allow creating output columns in script** checkbox is marked, a value of type `ConfigurableOutputColumnClass` is returned; otherwise, `OutputColumnClass` is returned. Both of these types are inherited from `ColumnClass` and implement the `Sequence` protocol (see Full API Description).

▼ Append

`Append()`

This method adds a new row to the output dataset. It has no arguments.

▼ Set

`Set(col, value)`

- `col` : The column index or name. Accepts values of type `int` or `str` .
- `value` : The value of the following types: `bool` , `int` , `float` , `str` , `datetime.datetime` , or `None` .

This method sets the value of the specified column in the row added by the `Append()` method.

▼ AssignColumns

`AssignColumns(array)`

- `array` : An iterable object containing elements of type `ColumnInfo` (see Full API Description). This method creates columns for the output dataset from a collection of `ColumnInfo` elements.

▼ AddColumn

`AddColumn(ColumnInfo, Name, DisplayName, DataType, DataKind, DefaultUsageType)`

Accepts arguments by keyword:

- `ColumnInfo` : A value of type `ColumnInfo` (see Full API Description). Optional argument.
- `Name` : The column name, a value of type `str` . Optional argument.
- `DisplayName` : The column label, a value of type `str` . Optional argument.
- `DataType` : The column data type, a value of type `int` (see Enumerations). Optional argument.
- `DataKind` : The column's data kind, a value of type `int` (see Enumerations). Optional argument.
- `DefaultUsageType` : Default usage of the column, a value of type `int` (see Enumerations). Optional argument.

This method inserts a column at the end of the output set's column list. It returns a value of type `OutputColumnClass` (see Full API Description).

▼ InsertColumn

`InsertColumn(Index, ColumnInfo, Name, DisplayName, DataType, DataKind, DefaultUsageType)`

Accepts arguments by keywords:

- `Index` : The index of the column in the column collection. Accepts a value of type `int` .
- `ColumnInfo` : A value of type `ColumnInfo` (see Full API Description). Optional argument.
- `Name` : The column name, a value of type `str` . Optional argument.
- `DisplayName` : The column label, a value of type `str` . Optional argument.
- `DataType` : The column's data type, a value of type `int` (see Enumerations). Optional argument.

- `DataKind` : The column's data kind, a value of type `int` (see Enumerations). Optional argument.
- `DefaultUsageType` : Default usage of the column, a value of type `int` (see Enumerations). Optional argument.

This method inserts a column at the specified index in the output dataset. It returns a value of type `OutputColumnClass` (see Full API Description).

▼ DeleteColumn

```
DeleteColumn(col)
```

- `col` : The column's index or name. Accepts values of type `int` or `str` .

This method deletes a column by its name or index.

▼ ClearColumns

```
ClearColumns()
```

This method clears the column list. It has no arguments.

Using the builtin_pandas_utils module

If the **Allow creating output columns in script** option is enabled, the following methods are available:

▼ prepare_compatible_table

```
prepare_compatible_table(table, dataframe, with_index)
```

This method sets the field structure of an `OutputTable` from a `pandas.DataFrame` . Arguments:

- `table` : A reference to the output `OutputTableClass` dataset.
- `dataframe` : A reference to the `pandas.DataFrame` whose structure is used to create the output dataset columns.
- `with_index` : If this argument is `True` , the `pandas.DataFrame` indexes are included in the structure of the output dataset. This is an optional argument. The default value is `False` .

▼ fill_table

```
fill_table(table, dataframe, with_index)
```

This method writes data from a `pandas.DataFrame` to an `OutputTable` . Arguments:

> - `table` : A reference to the output dataset. Accepts a value of type `OutputTableClass` .
> - `dataframe` : A reference to the `pandas.DataFrame` .
> - `with_index` : If this argument is `True` , the `pandas.DataFrame` indexes are exported to the output dataset. This is an optional argument. The default value is `False` .

You can find an example of using `builtin_pandas_utils` module below (see Example 2).

Examples

Example 1

```
from builtin_data import InputTable, OutputTable, DataType, DataKind,
UsageType
import datetime

#Copying columns from the input dataset
OutputTable.AssignColumns(InputTable.Columns)
#Deleting a column by index
OutputTable.DeleteColumn(0)
#Deleting a column by name
OutputTable.DeleteColumn("Test1")
#Deleting the entire list of columns
OutputTable.ClearColumns()
#Adding a column to the end of the output dataset's column list
OutputTable.AddColumn(Name="COL0",
                      DisplayName="Date/Time",
                      DataType=DataType.DateTime,
                      DataKind=DataKind.Continuous,
                      DefaultUsageType=UsageType.Active)
#Inserting a column at a specified index in the output dataset's column
list
OutputTable.InsertColumn(Index=0,
                         Name="COL1",
                         DisplayName="Flag",
                         DataType=DataType.Boolean)
#Getting a reference to a column by name
COL0 = OutputTable.GetColumn("COL0")
COL1 = OutputTable.GetColumn("COL1")
#Outputting the column property values
print("Index: ", COL1.Index)
print("Name: ", COL1.Name)
print("DisplayName: ", COL1.DisplayName)
print("DataType: ", COL1.DataType)
print("DataKind: ", COL1.DataKind)
print("DefaultUsageType: ", COL1.DefaultUsageType)

#Adding a row to the output dataset
OutputTable.Append()
```

```
#The current Date/Time is written to the field with index 0
OutputTable.Columns[1].Set(datetime.datetime.now())
#The value "true" is written to the field with index 1
OutputTable.GetColumn(0).Set(True)
#Copying the values of the first row to the second
OutputTable.Append()
for i in range(OutputTable.ColumnCount):
    value = OutputTable.Get(0, i)
    OutputTable.Set(i, value)

#Checking that the value in the row with index 0 in the column with index
1 is not defined
print(OutputTable.IsNull(0, 1))
print(OutputTable.Get(0, 1) is None)

print("RowCount = ", OutputTable.RowCount)
#Output: RowCount =  2
```

Example 2

Using the `builtin_pandas_utils` module.

```
from builtin_data import InputTable, OutputTable,
ConfigurableOutputTableClass
from builtin_pandas_utils import to_data_frame, prepare_compatible_table,
fill_table

#The input port is optional and may not contain data
if InputTable:
    #Create a pd.DataFrame from the input dataset
    input_frame = to_data_frame(InputTable)
    #Grouping input_frame
    output_frame = input_frame.groupby(["Class"]).sum()
    #If the "Allow creating output columns in script" option is enabled,
    #the structure of the output dataset can be prepared from a
pd.DataFrame
    assert isinstance(OutputTable, ConfigurableOutputTableClass)
    #Defining the structure of the output dataset
    prepare_compatible_table(OutputTable, output_frame, with_index=True)
    #Filling the output dataset
    fill_table(OutputTable, output_frame, with_index=True)
```

Read on: Python: Enumerations