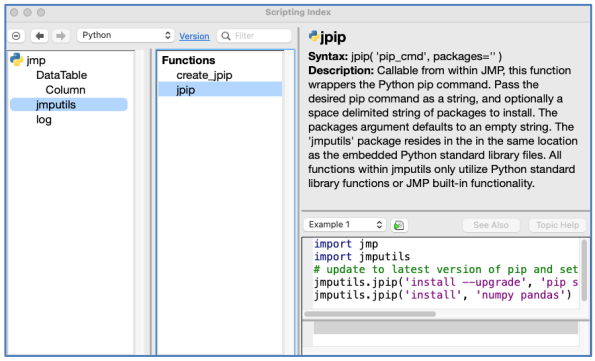
Integrating JMP and Python

JMP can integrate with Python, combining JMP's capabilities and interactive interface with Python's additional tools. Use cases include: running a Python script directly on a JMP data table; using Python to import unsupported file types into JMP; blending Python and JMP Scripting Language (JSL) code together to combine their capabilities; creating a JMP add-in that leverages Python; using Python as an intermediary between JMP and other software.

Note: As of JMP 18, JMP includes a single, supported version of Python, an extensive set of Python libraries, a robust connection between JMP and Python data structures, and a built-in Python script editor. JMP's Python environment includes two packages – jmp and jmputils – that support JMP and Python integration. Go to **Help > Scripting Index** and select **Python** in the drop-down menu to learn about these packages, including the functions highlighted below. Also see JSL's Python integration functions for calling out to Python from a JSL script.



- `jmputils.jpip()`: install Python packages
- `jmp.current()`: returns JMP data table object for the currently active JMP data table
- `jmp.table()`: returns JMP data table object for a named JMP data table
- `jmp.DataTable()`: create a new JMP data table
- `jmp.run_jsl()`: run JSL code from within the Python script

An Example: Running a Python analysis on a JMP data table and returning results to JMP

Use **File > New > New Python Script** to create a new Python script in JMP. Download *jmp-and-python-integration-example.py* from the Learning Library page for an example script that opens a JMP data table, runs a Python analysis (Isolation Forest) on that data table, returns the result to the JMP data table, and uses JSL to set table row colors. Open and run the script with JMP.

Below is a summary of select lines in the example. See the full script for more code and comments.

- Line 7 installs the necessary Python packages into JMP's dedicated Python environment

```
7  jmputils.jpip('install', 'numpy pandas scikit-learn')
```

- Line 19 opens *Water Treatment.jmp* from JMP's sample data folder; `dt` is JMP data table object

```
19  dt = jmp.open(jmp.SAMPLE_DATA + 'Water Treatment.jmp')
```

- Lines 29-34 convert `dt` to a Pandas data frame for analysis; click the plus to expand lines 29-33

```
29 ⊞def jmpdt_to_pandasdf(jmpdt):
34  X = jmpdt_to_pandasdf(dt)
```

- Lines 44-46 return isolation forest anomaly labels as a new column in *Water Treatment.jmp*

```
44  dt.new_column('Anomaly', jmp.DataType.Numeric, jmp.ModelingType.Nominal)
45  labelsList = labels.tolist()
46  dt['Anomaly'] = labelsList
```

- Lines 50-53 use JSL to color the rows in *Water Treatment.jmp* by anomaly label

```
50  jmp.run_jsl('''
51  dt = Python Get( dt );
52  dt << Color or Mark by Column( :Anomaly, Color Theme( "JMP Default" ) );
53  ''')
```

Visit **"Scripting Guide > Python"** in **JMP Help** to learn more.