Highlight Note

Working with Tabular Datasets

You can read data directly from a tabular dataset by converting it into a Pandas or Spark dataframe:

```
df = tab_ds.to_pandas_dataframe()
# code to work with dataframe goes here, for example:
print(df.head())
```


Passing a Tabular Dataset to an Experiment Script

When you need to access a dataset in an experiment script, you must pass the dataset to the script. There are two ways you can do this.

Use a Script Argument

You can pass a tabular dataset as a script argument. When you take this approach, the argument received by the script is the unique ID for the dataset in your workspace. In the script, you can then get the workspace from the run context and use it to retrieve the dataset by it's ID.

ScriptRunConfig

```
env = Environment('my_env')
packages = CondaDependencies.create(conda_packages=['pip'],
                                    pip_packages=['azureml-defaults',
                                                  'azureml-dataprep[pandas]'])
env.python.conda_dependencies = packages

script_config = ScriptRunConfig(source_directory='my_dir',
                                script='script.py',
                                arguments=['--ds', tab_ds],
                                environment=env)
```

Script

```
from azureml.core import Run, Dataset

parser.add_argument('--ds', type=str, dest='dataset_id')
args = parser.parse_args()

run = Run.get_context()
ws = run.experiment.workspace
dataset = Dataset.get_by_id(ws, id=args.dataset_id)
data = dataset.to_pandas_dataframe()
```

Use a Named Input

Alternatively, you can pass a tabular dataset as a *named input*. In this approach, you use the **as_named_input** method of the dataset to specify a name for the dataset. Then in the script, you can retrieve the dataset by name

from the run context's **input_datasets** collection without needing to retrieve it from the workspace. Note that if you use this approach, you still need to include a script argument for the dataset, even though you don't actually use it to retrieve the dataset.

ScriptRunConfig

```
env = Environment('my_env')
packages = CondaDependencies.create(conda_packages=['pip'],
                                    pip_packages=['azureml-defaults',
                                                  'azureml-dataprep[pandas]'])
env.python.conda_dependencies = packages

script_config = ScriptRunConfig(source_directory='my_dir',
                                script='script.py',
                                arguments=['--ds', tab_ds.as_named_input('my_dataset')],
                                environment=env)
```

Script

```
from azureml.core import Run

parser.add_argument('--ds', type=str, dest='ds_id')
args = parser.parse_args()

run = Run.get_context()
dataset = run.input_datasets['my_dataset']
data = dataset.to_pandas_dataframe()
```