Highlight Note

Working with File Datasets

When working with a file dataset, you can use the **to_path()** method to return a list of the file paths encapsulated by the dataset:

```
for file_path in file_ds.to_path():
    print(file_path)
```

Passing a File Dataset to an Experiment Script

Just as with a Tabular dataset, there are two ways you can pass a file dataset to a script. However, there are some key differences in the way that the dataset is passed.

Use a Script Argument

You can pass a file dataset as a script argument. Unlike with a tabular dataset, you must specify a mode for the file dataset argument, which can be **as_download** or **as_mount**. This provides an access point that the script can use to read the files in the dataset. In most cases, you should use **as_download**, which copies the files to a temporary location on the compute where the script is being run. However, if you are working with a large amount of data for which there may not be enough storage space on the experiment compute, use **as_mount** to stream the files directly from their source.

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', file_ds.as_download()],
                                environment=env)
```

Script

```
from azureml.core import Run
import glob

parser.add_argument('--ds', type=str, dest='ds_ref')
args = parser.parse_args()
run = Run.get_context()

imgs = glob.glob(ds_ref + "/*.jpg")
```

Use a Named Input

You can also pass a file dataset as a *named input*. In this approach, you use the **as_named_input** method of the dataset to specify a name before specifying the access mode. Then in the script, you can retrieve the dataset by name from the run context's **input_datasets** collection and read the files from there. As with tabular datasets, if you use a named input, 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',
file_ds.as_named_input('my_ds').as_download()],
                                environment=env)
```

Script

```
from azureml.core import Run
import glob

parser.add_argument('--ds', type=str, dest='ds_ref')
args = parser.parse_args()
run = Run.get_context()

dataset = run.input_datasets['my_ds']
imgs= glob.glob(dataset + "/*.jpg")
```