Highlight Note

Running a Script as an Experiment

You can run an experiment inline using the **start_logging** method of the **Experiment** object, but it's more common to encapsulate the experiment logic in a script and run the script as an experiment. The script can be run in any valid compute context, making this a more flexible solution for running experiments at scale.

Writing an Experiment Script

An experiment script is just a Python code file that contains the code you want to run in the experiment. To access the experiment run context (which is needed to log metrics) the script must import the **azureml.core.Run** class and call its **get_context** method. The script can then use the run context to log metrics, upload files, and complete the experiment, as shown here:

```
from azureml.core import Run
import pandas as pd
import matplotlib.pyplot as plt
import os

# Get the experiment run context
run = Run.get_context()

# load the diabetes dataset
data = pd.read_csv('data.csv')

# Count the rows and log the result
row_count = (len(data))
run.log('observations', row_count)

# Save a sample of the data
os.makedirs('outputs', exist_ok=True)
data.sample(100).to_csv("outputs/sample.csv", index=False, header=True)

# Complete the run
run.complete()
```

The experiment script is saved in a folder along with any other files on which it depends. For example, you could save this script as **experiment.py** in a folder named **experiment_folder**. Since the script includes code to load training data from **data.csv**, this file should also be saved in the folder.

Running an Experiment Script

To run a script as an experiment, you must define a *script configuration* that defines the script to be run and the Python environment in which to run it. This is implemented by using a **ScriptRunConfig** object.

For example, the following code could be used to run an experiment based on a script in the **experiment_files** folder (which must also contain any files used by the script, such as the *data.csv* file in previous script code example):

```
from azureml.core import Experiment, ScriptRunConfig

# Create a script config
script_config = ScriptRunConfig(source_directory='experiment_folder',
                                script='experiment.py')

# submit the experiment
experiment = Experiment(workspace=ws, name='my-experiment')
run = experiment.submit(config=script_config)
run.wait_for_completion(show_output=True)
```

Note: An implicitly created **RunConfiguration** object defines the Python environment for the experiment, including the packages available to the script. The default environment includes standard Python libraries like **numpy** and **pandas** as well as the **azureml-defaults** package, which contains the classes needed to work with the experiment run context. If your script depends on packages that are not included in the default environment, you must associate the **ScriptRunConfig** with an **Environment** object that makes use of a **CondaDependencies** object to specify the Python packages required. Runtime environments are discussed in more detail later in this course.