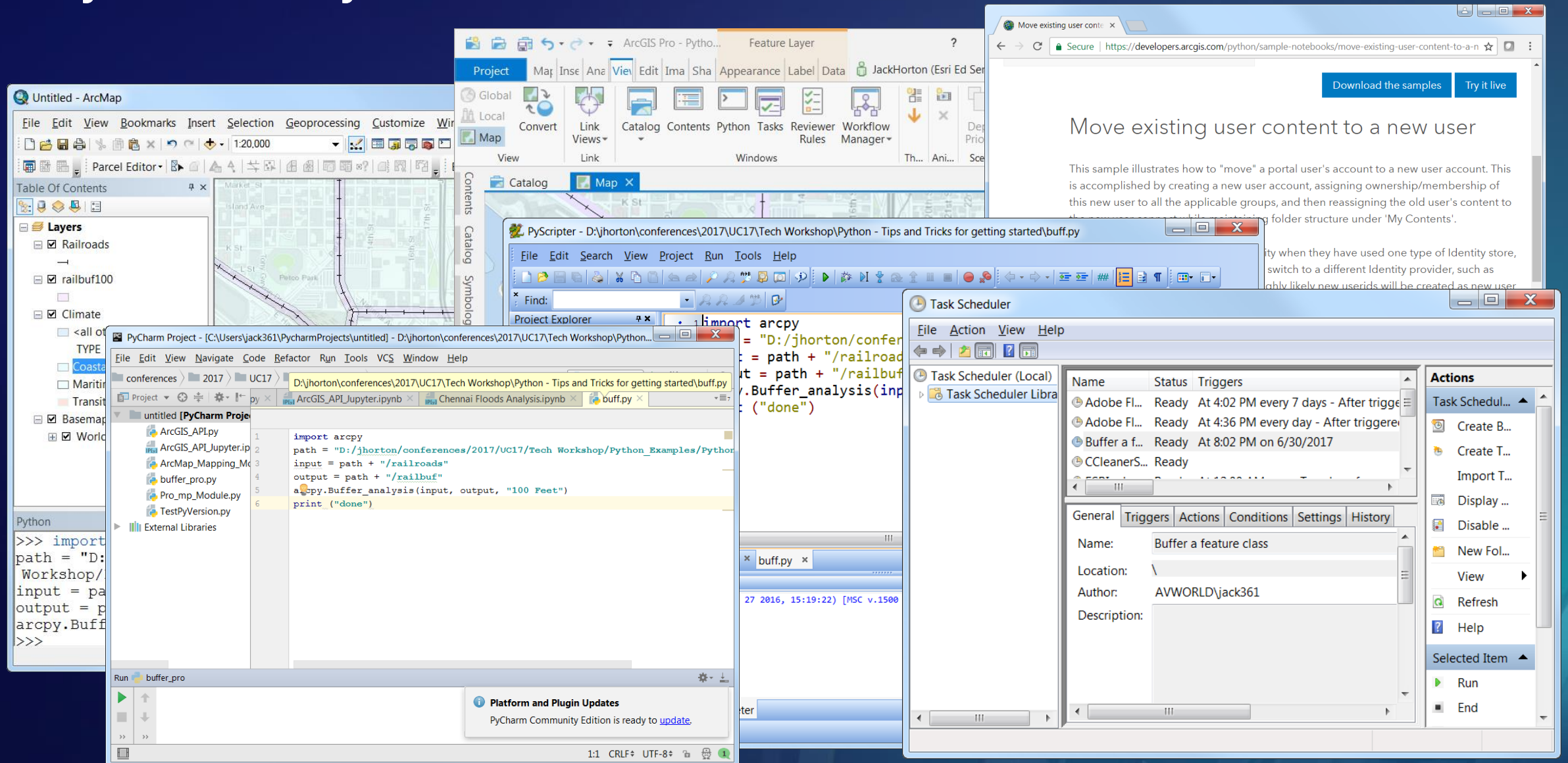
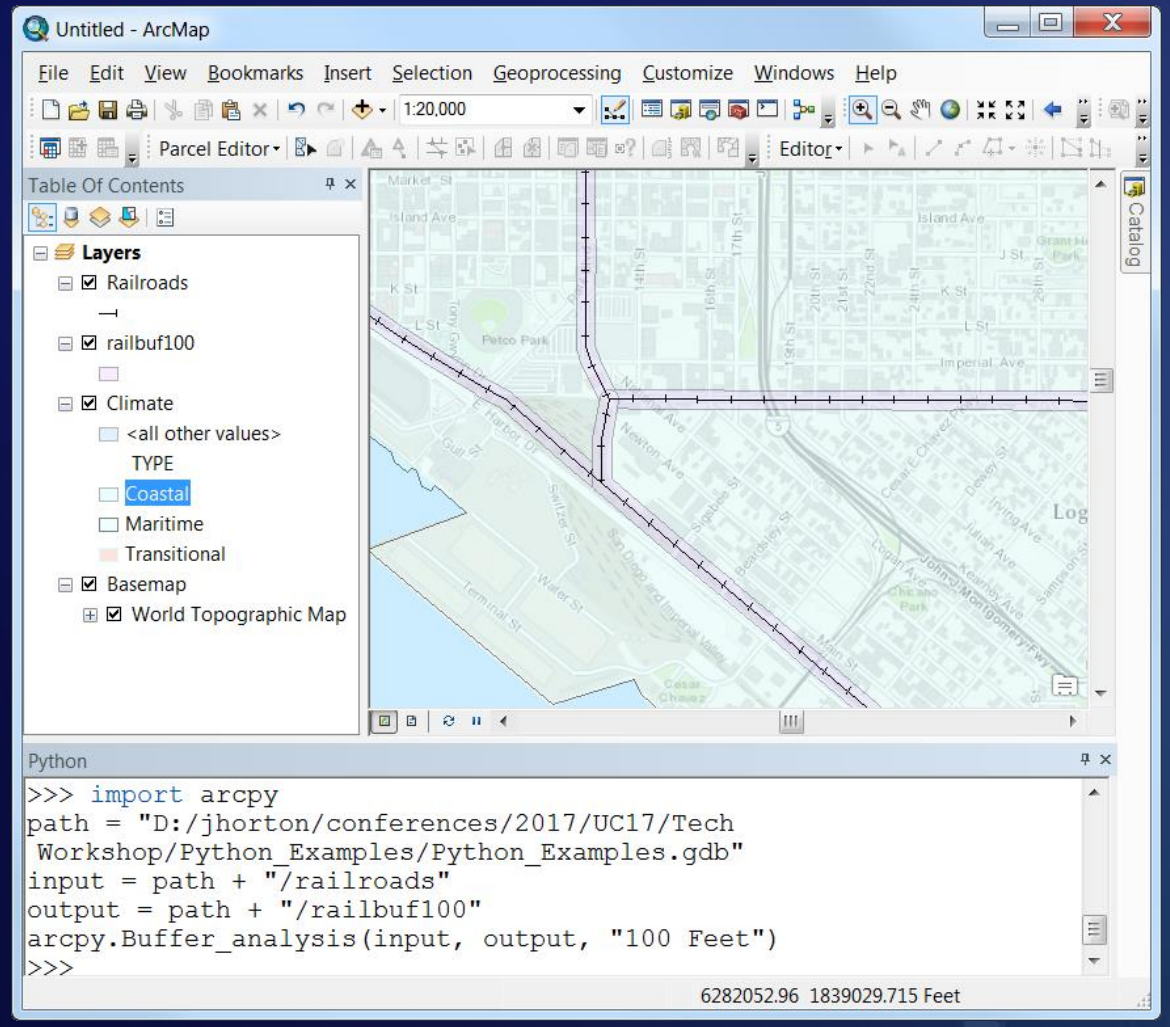
Python: Top 5 Tips and Tricks

Jack Horton

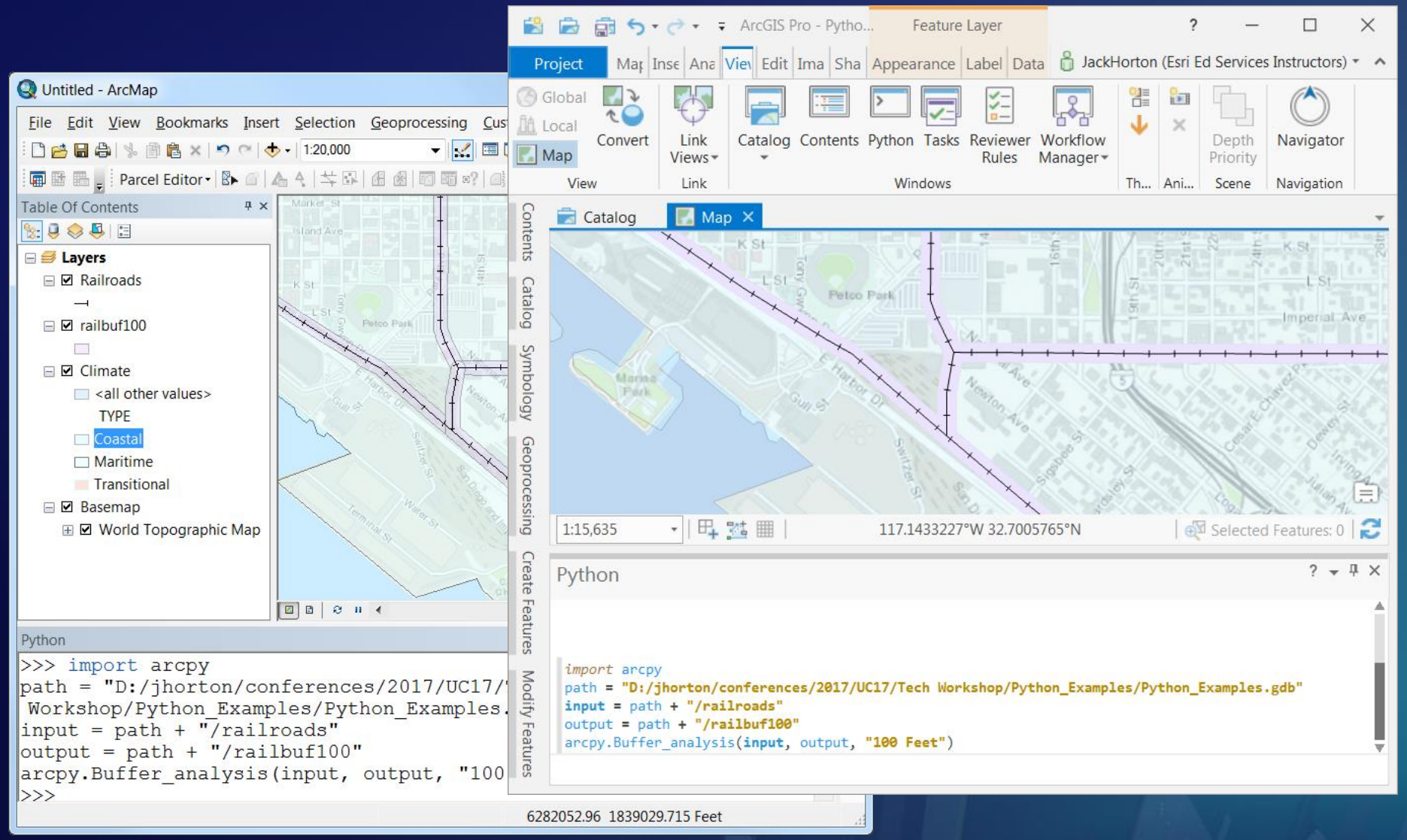
Python is everywhere in ArcGIS



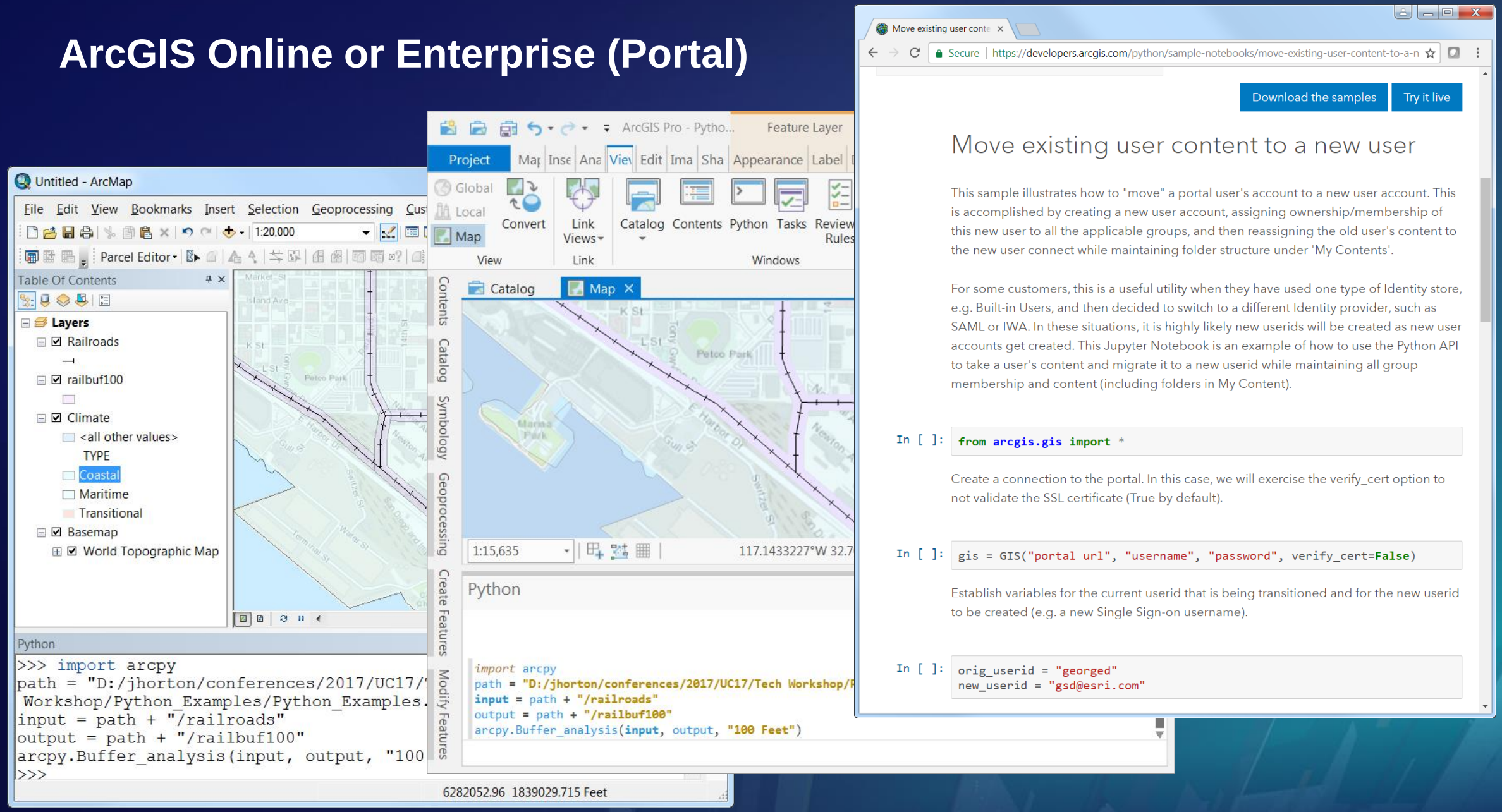
ArcMap


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:20,000
Parcel Editor
Editor
Table Of Contents
Layers
Railroads
railbuf100
Climate
<all other values>
TYPE
Coastal
Maritime
Transitional
Basemap
World Topographic Map
Catalog
Python
>>> import arcpy
path = "D:/jhorton/conferences/2017/UC17/Tech
Workshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf100"
arcpy.Buffer_analysis(input, output, "100 Feet")
>>>
6282052.96 1839029.715 Feet

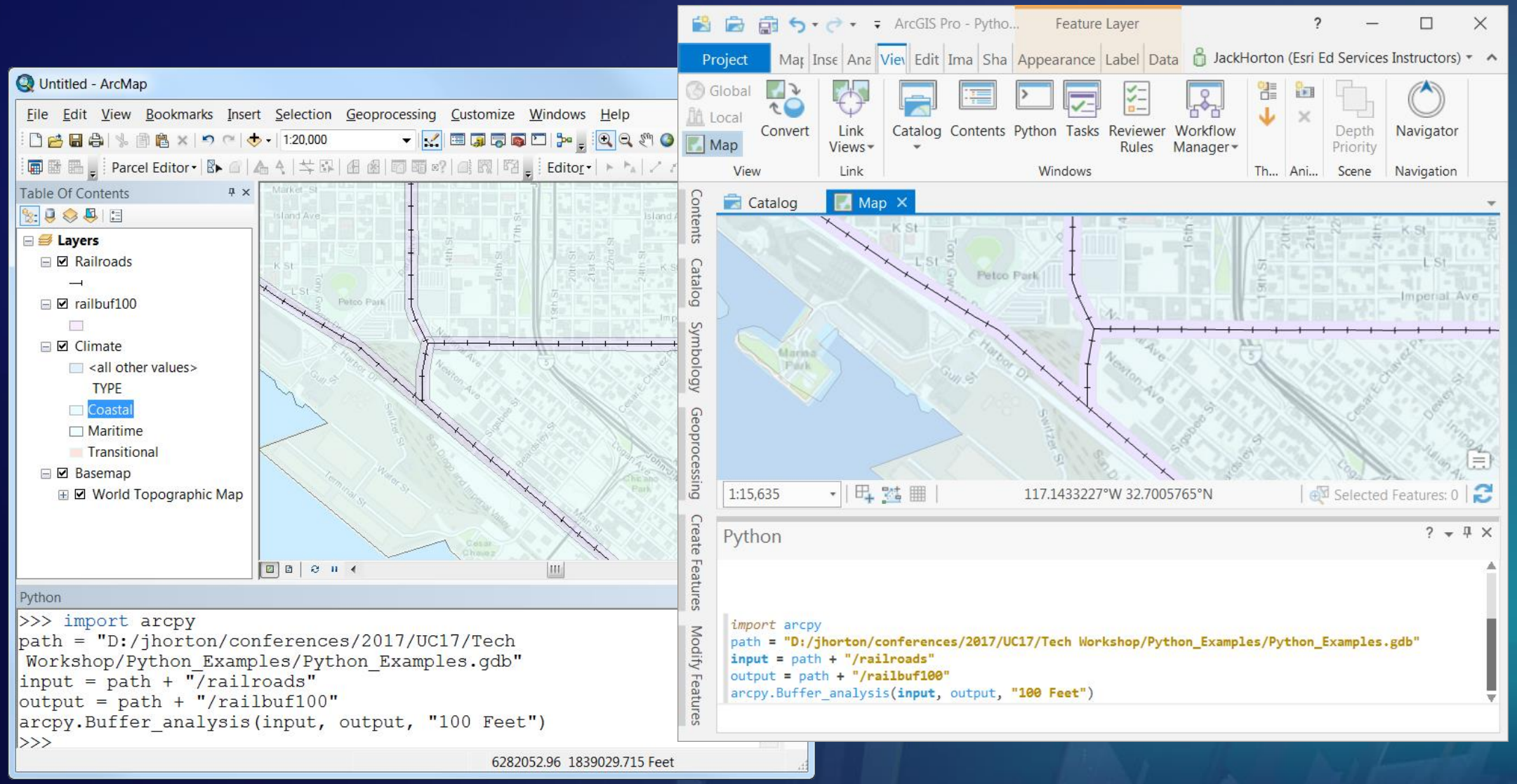
ArcGIS Pro


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing
Table Of Contents
Layers
Railroads
railbuf100
Climate
<all other values>
TYPE
Coastal
Maritime
Transitional
Basemap
World Topographic Map
Python
>>> import arcpy
path = "D:/jhorton/conferences/2017/UC17/
Workshop/Python_Examples/Python_Examples.
input = path + "/railroads"
output = path + "/railbuf100"
arcpy.Buffer_analysis(input, output, "100
>>>
6282052.96 1839029.715 Feet
ArcGIS Pro - Pytho...
Feature Layer
Project
JackHorton (Esri Ed Services Instructors)
Global
Local
Map
Convert
Link Views
Catalog
Contents
Python
Tasks
Reviewer Rules
Workflow Manager
Depth Priority
Navigator
View
Link
Windows
Scene
Navigation
Catalog
Map
Contents
Catalog
Symbology
Geoprocessing
Create Features
Modify Features
1:15,635
117.1433227°W 32.7005765°N
Selected Features: 0
Python
import arcpy
path = "D:/jhorton/conferences/2017/UC17/Tech Workshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf100"
arcpy.Buffer_analysis(input, output, "100 Feet")

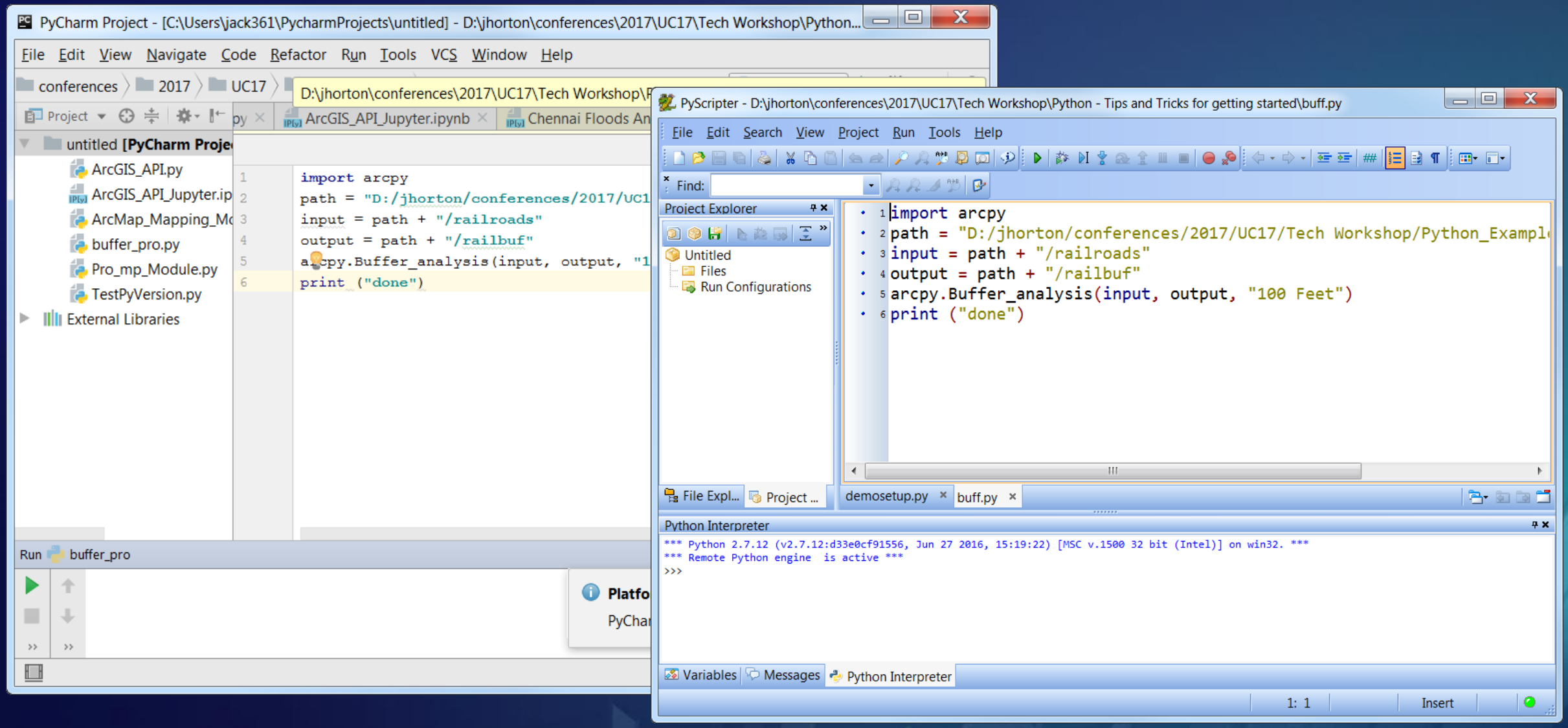
ArcGIS Online or Enterprise (Portal)



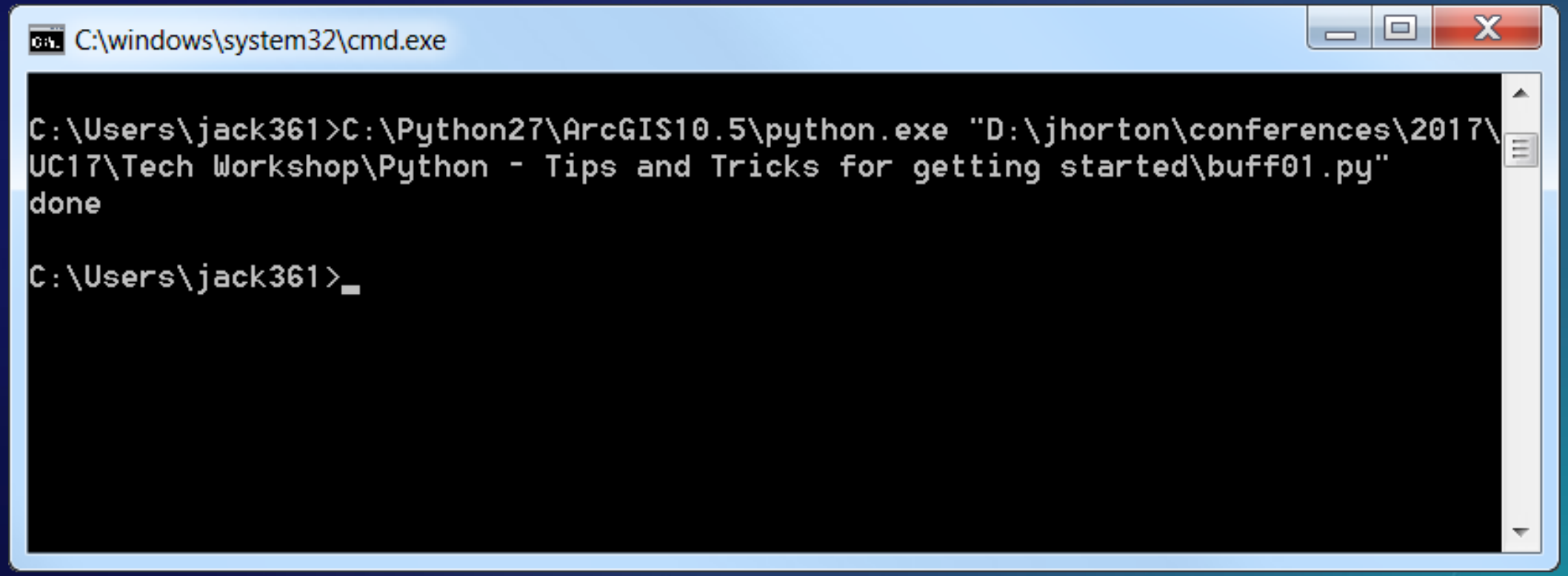
You can run it in the application



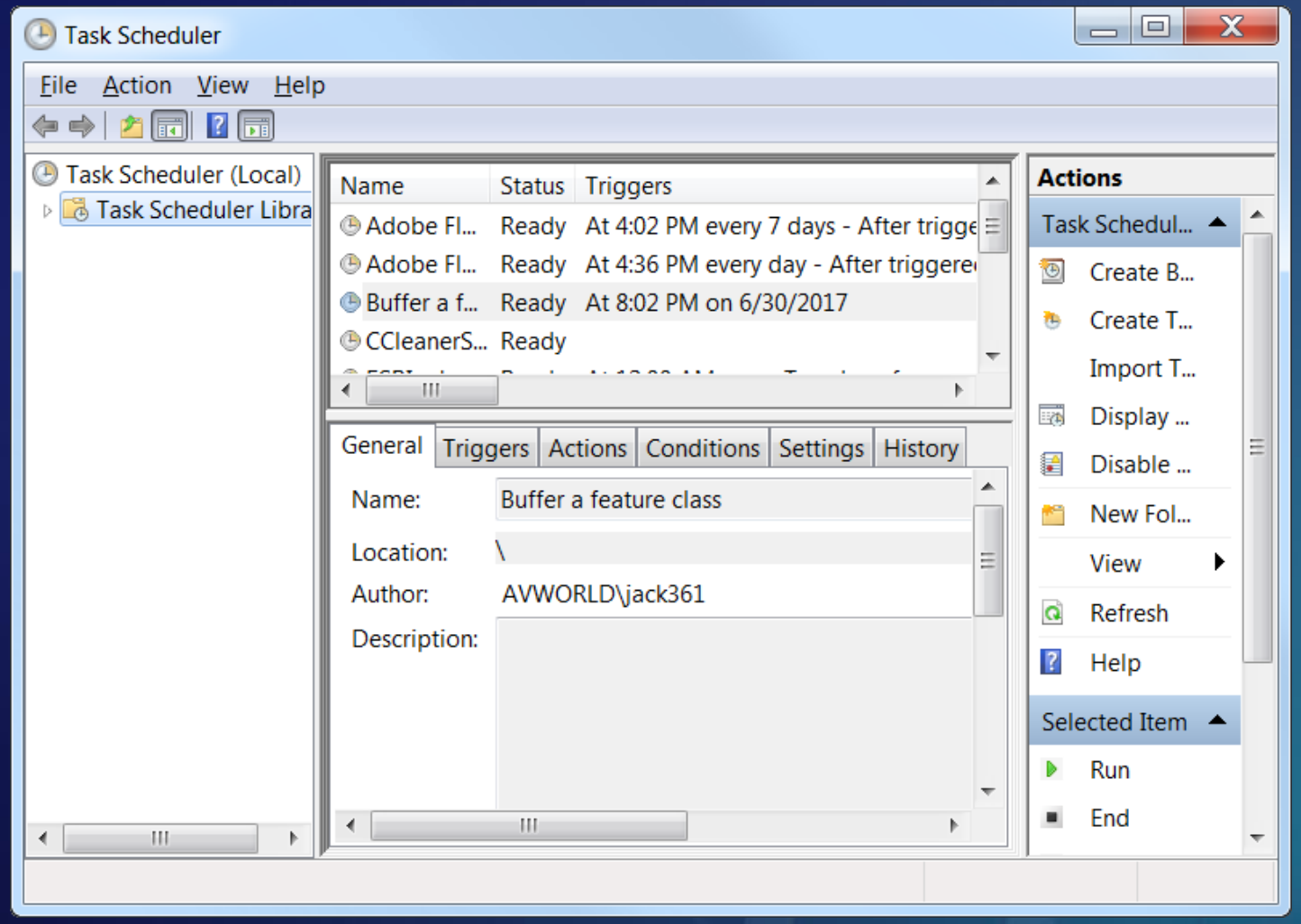
You can run it in an IDE (Interactive Development Environment)



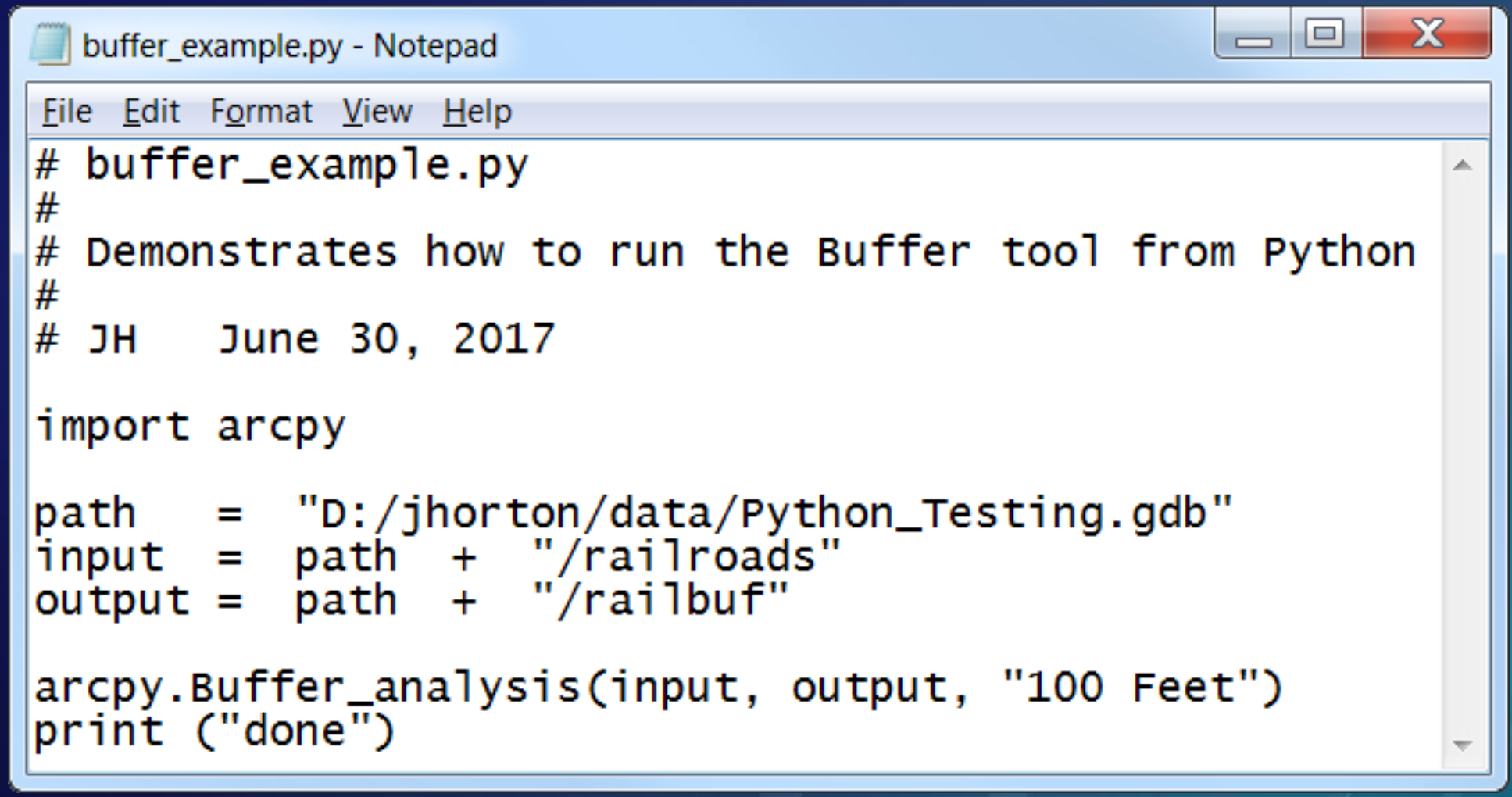
You can run it standalone



Or you can schedule it to run automatically



A Python script is just a text file



Python automates the things you do by hand, such as

- Mapping
- Data management
- Analysis
- Publishing web services
- Administering your portal
- And much more

In this workshop, we will focus on simple analysis running geoprocessing tools.

The Help is full of code samples

Code sample

Buffer example 1 (Python window)

The following Python window script demonstrates how to use the Buffer tool.

```
import arcpy
arcpy.env.workspace = "C:/data"
arcpy.Buffer_analysis("roads", "C:/output/majorrdsBuffered", "100 Feet", "FULL", "ROUND", "LIST", "Distance")
```

…Copy and paste them to get started

arcpy contains the Esri Python code in ArcGIS Pro and ArcMap

We import the arcpy module so we can get to things like the buffer tool

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Variables store data

In this script, we put a long pathname in a variable called path

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Variables store data

Then we use it to make two new variables containing the pathnames to our data

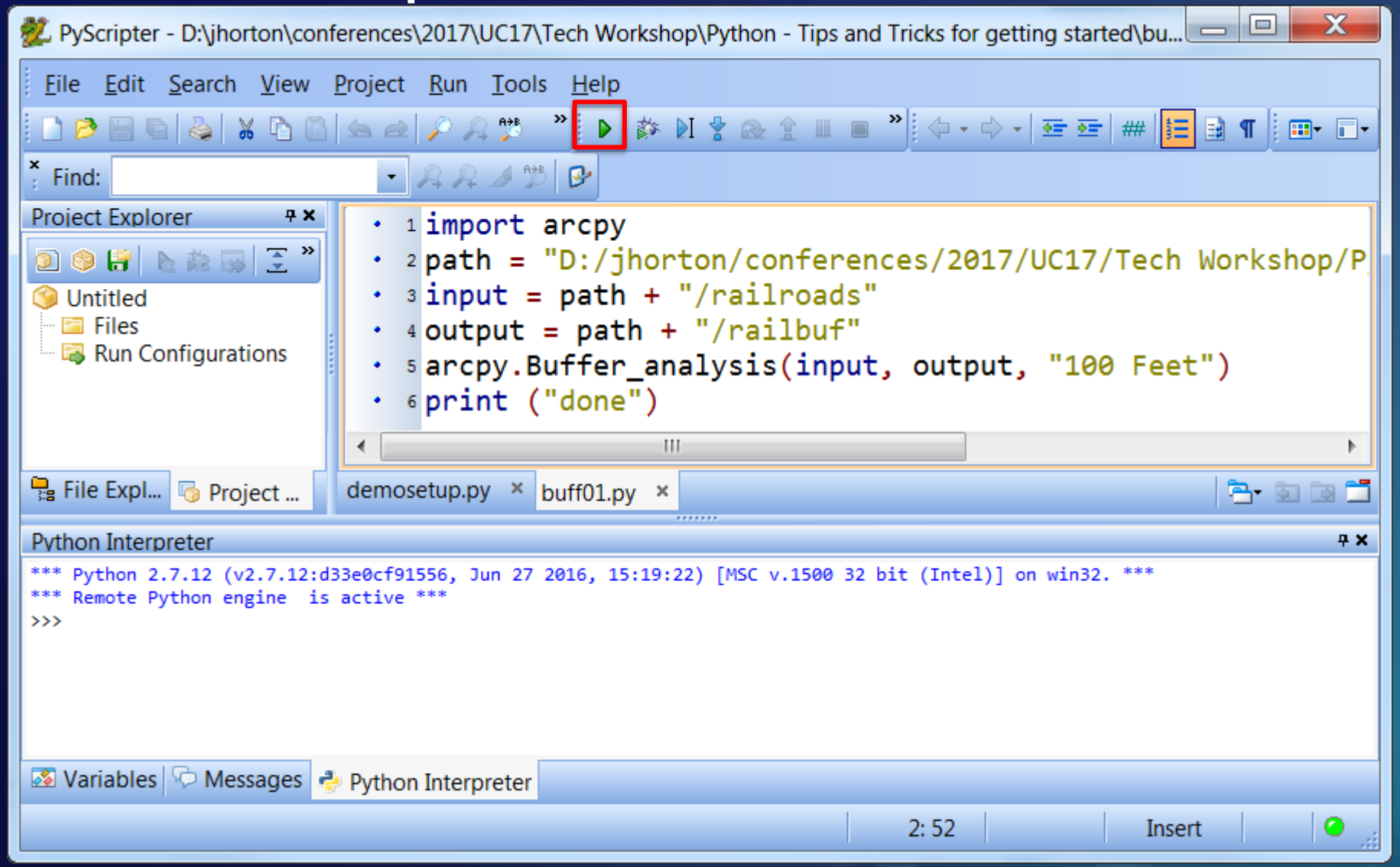
```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Variables store data

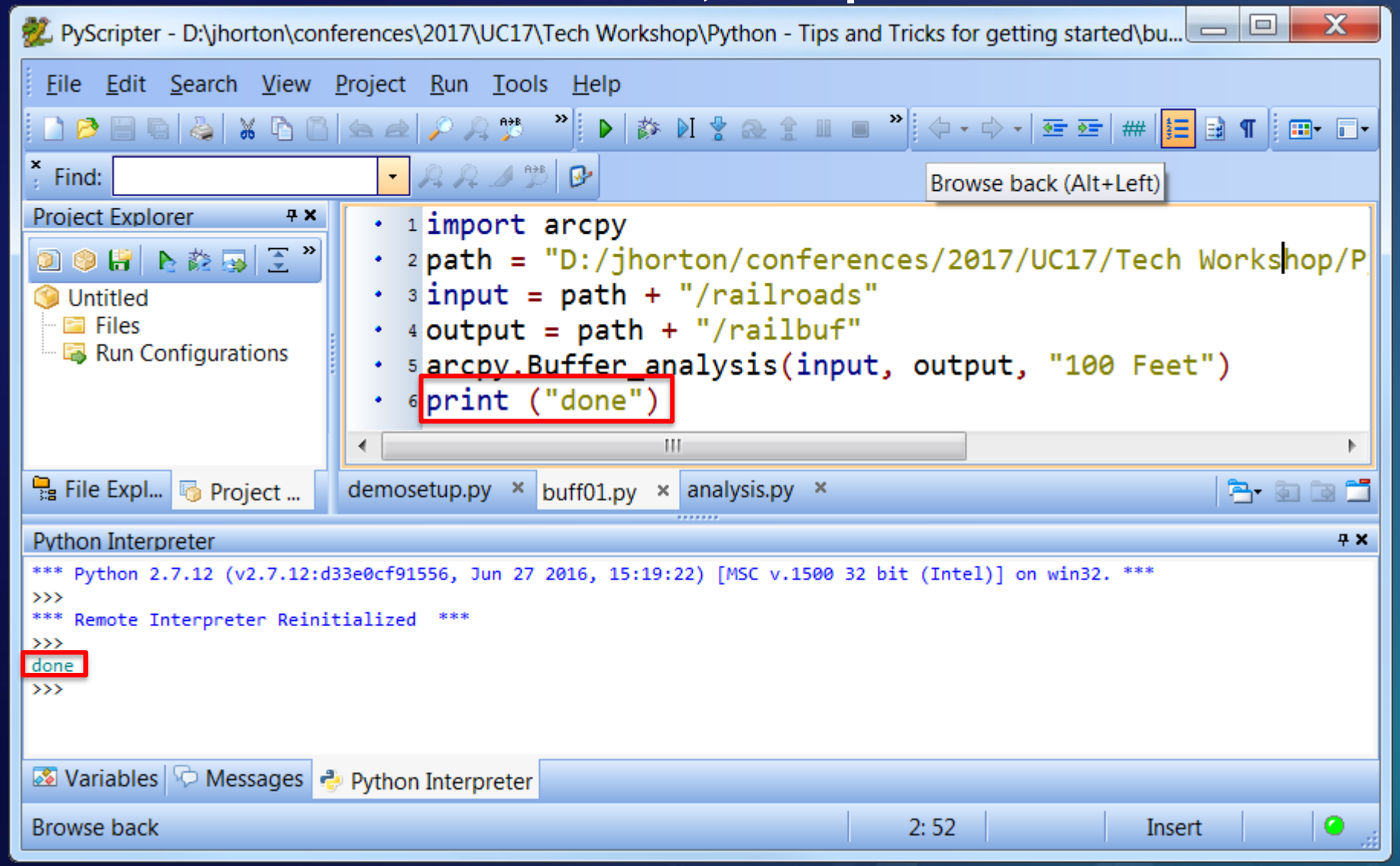
And we finally use these two variables as input to the Buffer geoprocessing tool

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

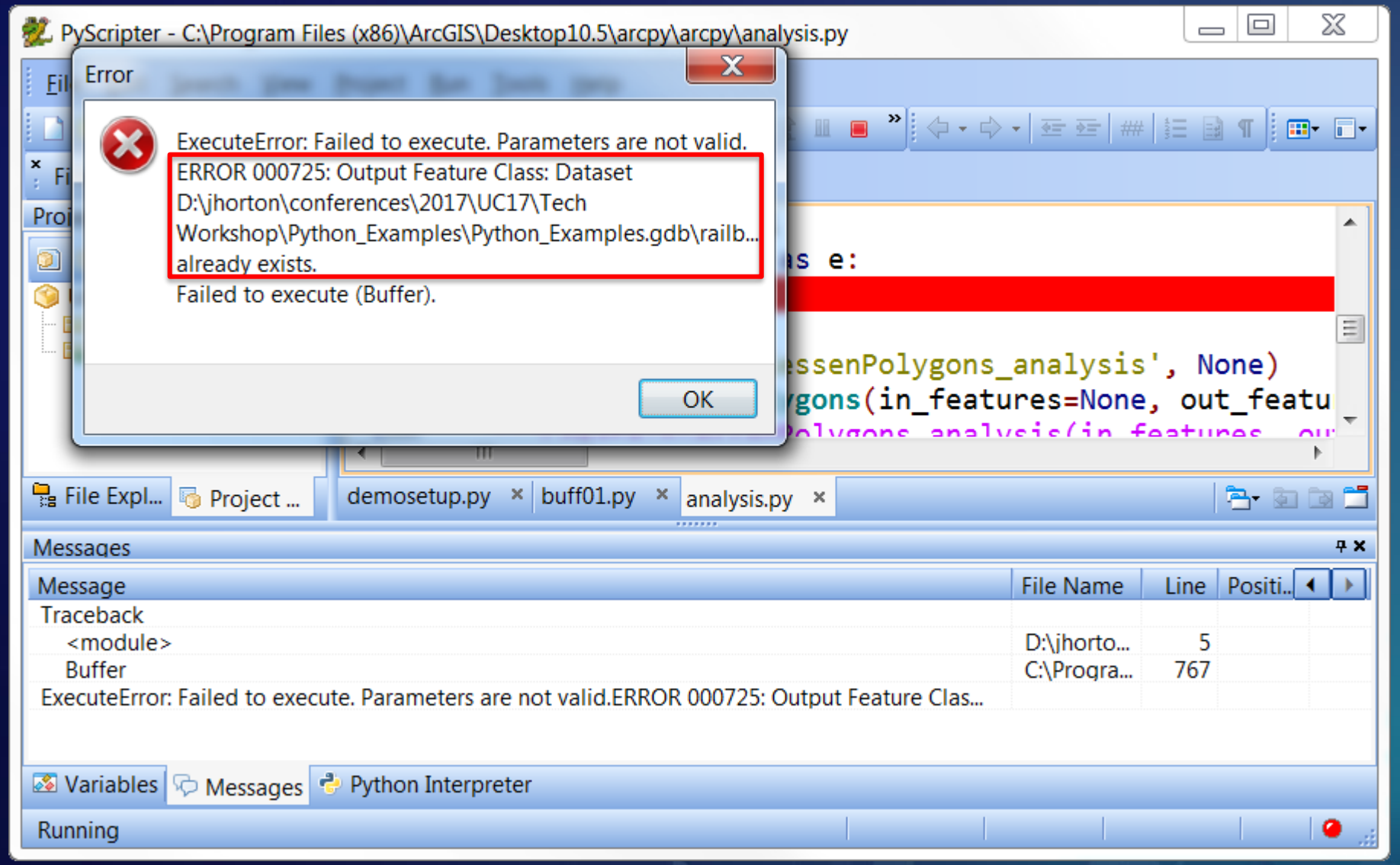
Let's run our script



It worked! – It made the buffer, then printed “done”

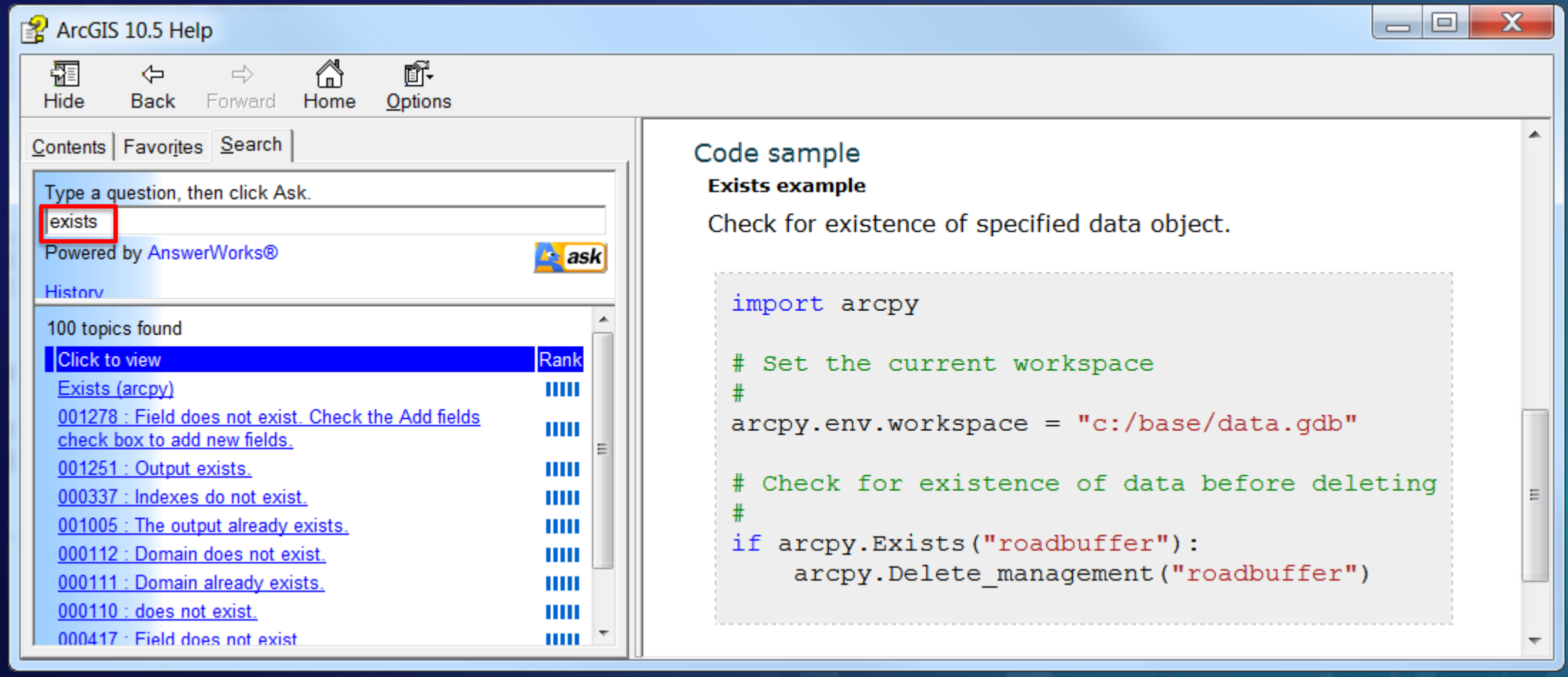


Lets run it again – it fails, because the output feature class already exists



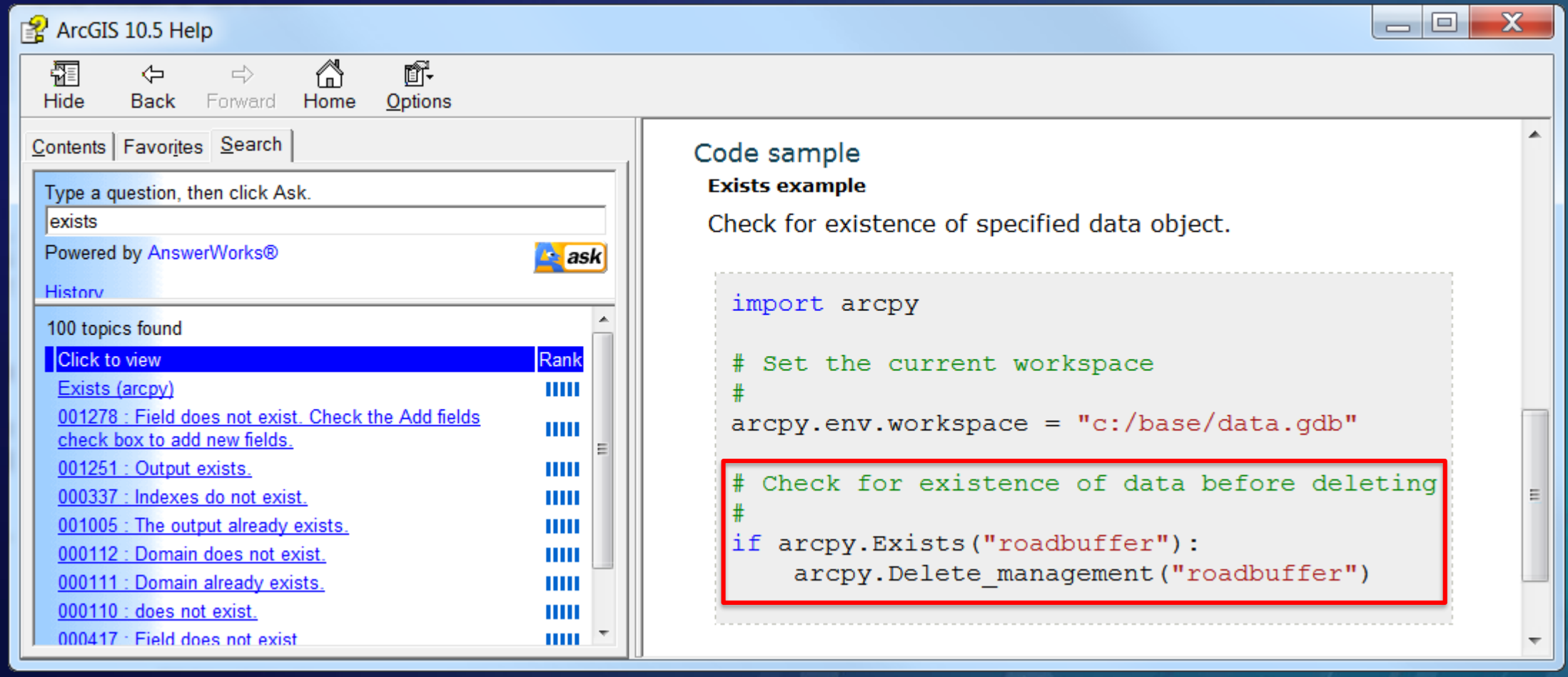
Let’s delete it if it already exists

- Search for “Exists” and get another code sample from the help



Let’s delete it if it already exists

- Search for “Exists” and get another code sample from the help



Lets follow the pattern in the sample and put it in our code

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Lets follow the pattern in the sample and put it in our code

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"


arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Lets follow the pattern in the sample and put it in our code

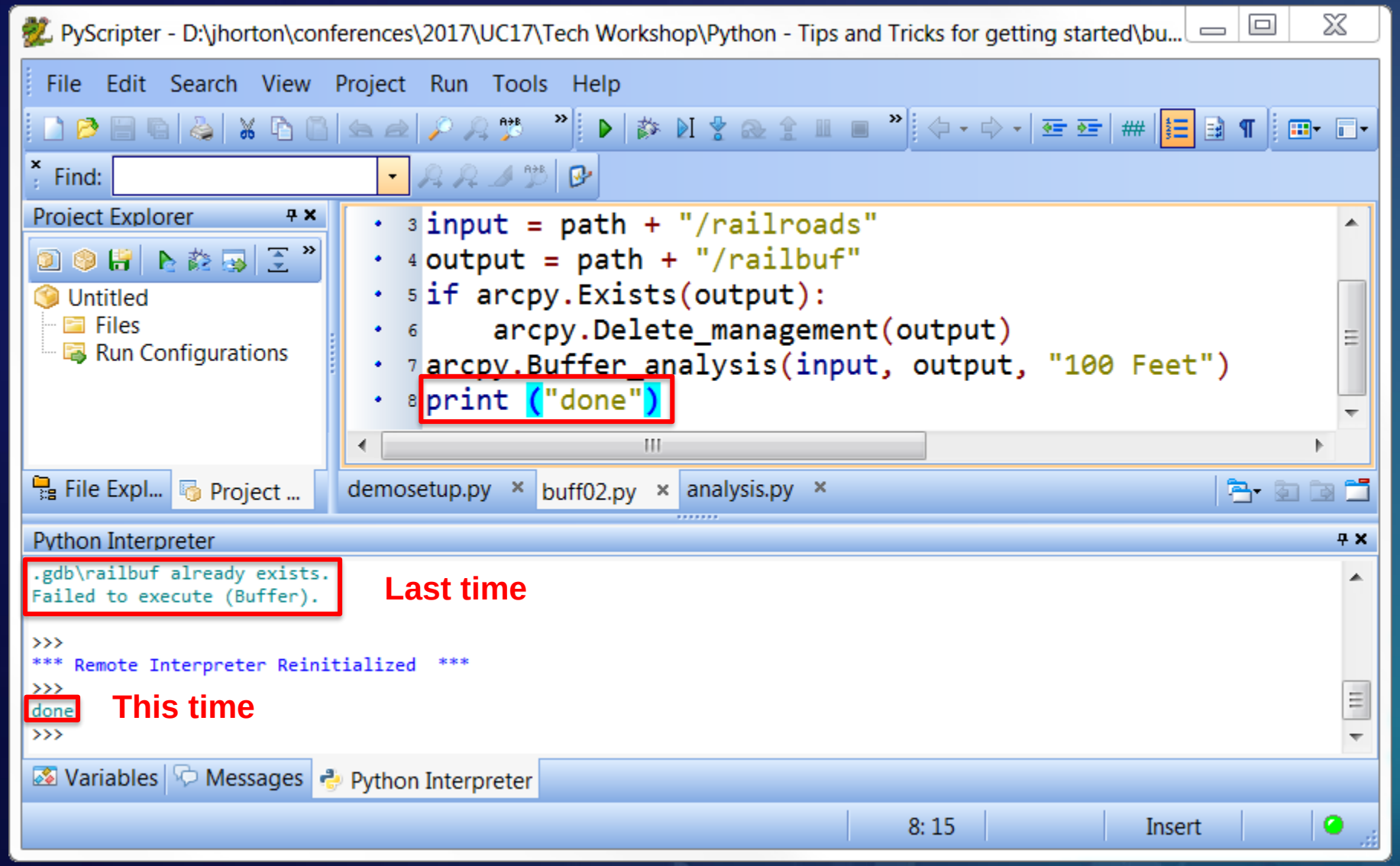
```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
if arcpy.Exists(output):
    arcpy.Delete_management(output)
arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Lets follow the pattern in the sample and put it in our code

Note that we don't use quotes, because our quoted string is in the variable called output

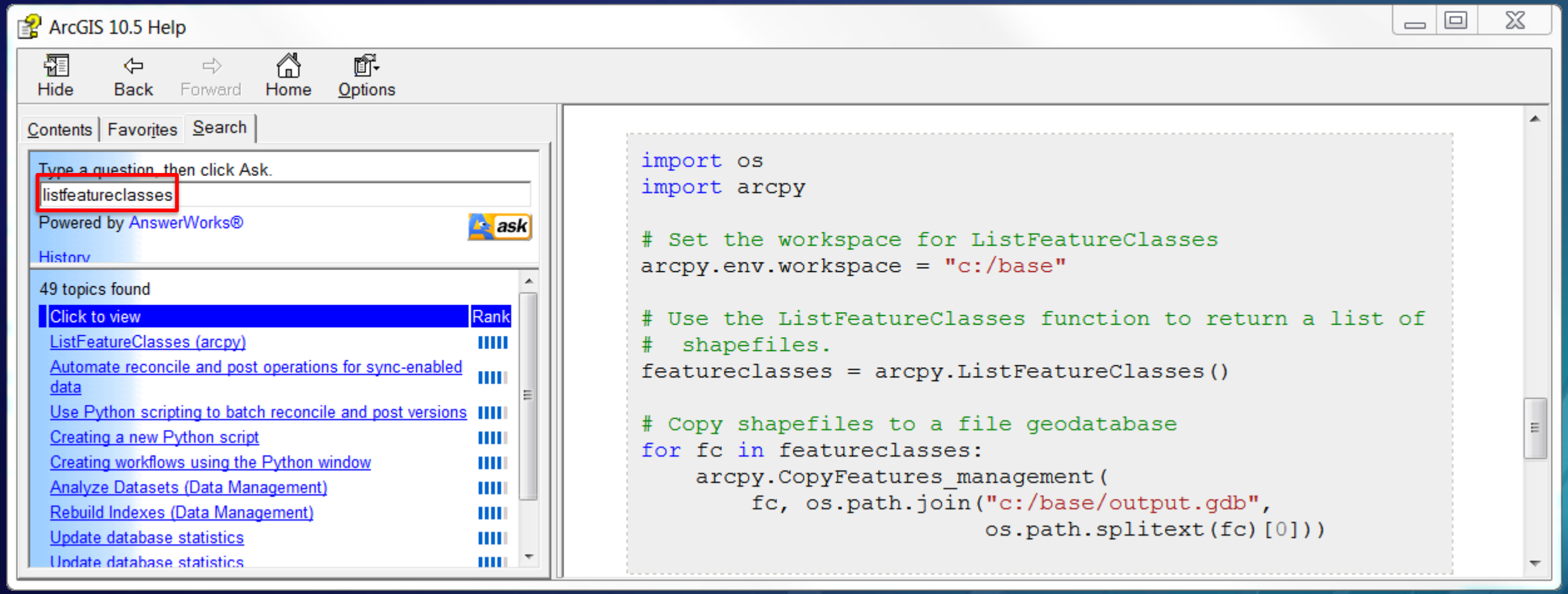
```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
if arcpy.Exists(output):
    arcpy.Delete_management(output)
arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Now it works, because it deletes the output feature class if it already exists



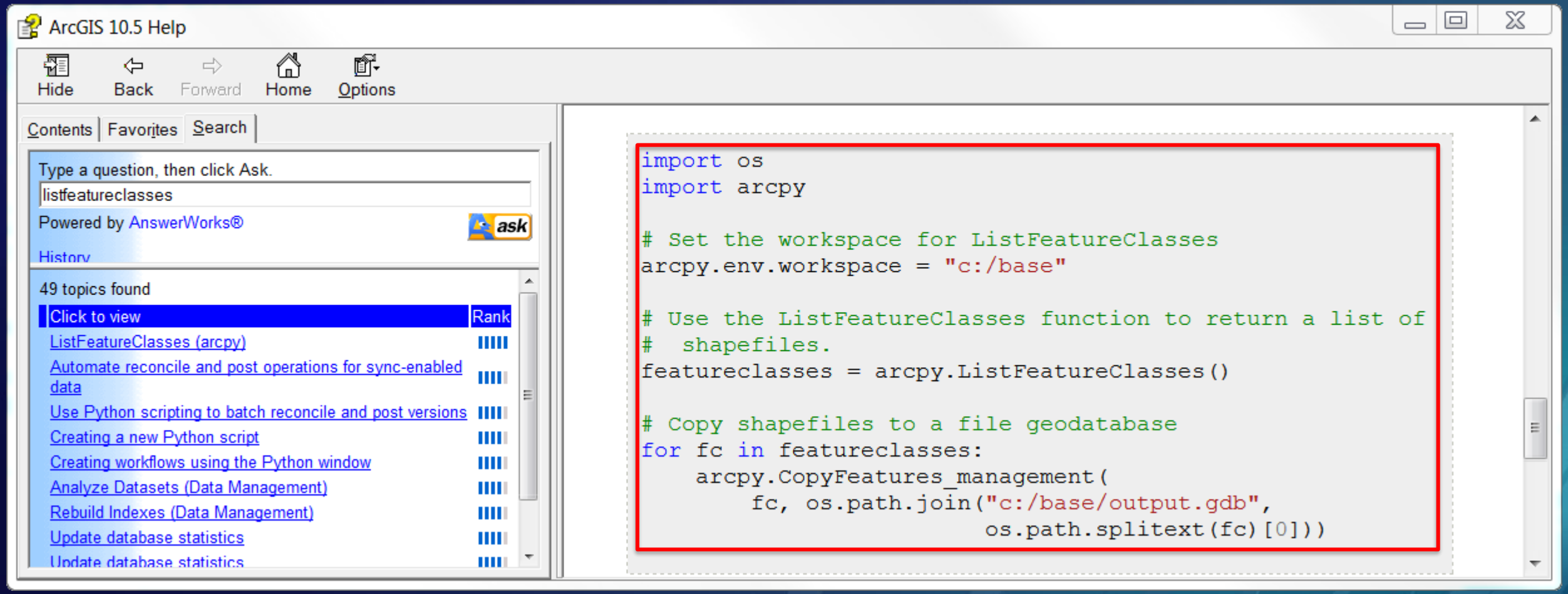
Lets buffer all the feature classes in the geodatabase

Search the help for ListFeatureClasses



Lets buffer all the feature classes in the geodatabase

And copy the sample



This sample is kind of fancy – lets remove what we don't need

```
import os
import arcpy

# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"

# Use the ListFeatureClasses function to return a list of
#  shapefiles.
featureclasses = arcpy.ListFeatureClasses()

# Copy shapefiles to a file geodatabase
for fc in featureclasses:
    arcpy.CopyFeatures_management(
        fc, os.path.join("c:/base/output.gdb",
                         os.path.splitext(fc)[0]))
```

This sample is kind of fancy – lets remove what we don't need

This example uses a Standard Python Module called OS to manipulate pathnames

```
import os
import arcpy

# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"

# Use the ListFeatureClasses function to return a list of
#  shapefiles.
featureclasses = arcpy.ListFeatureClasses()

# Copy shapefiles to a file geodatabase
for fc in featureclasses:
    arcpy.CopyFeatures_management(
        fc, os.path.join("c:/base/output.gdb",
                         os.path.splitext(fc)[0]))
```

This sample is kind of fancy – lets just pull out what we need

Lets remove that part…

```
import arcpy

# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"

# Use the ListFeatureClasses function to return a list of
#  shapefiles.
featureclasses = arcpy.ListFeatureClasses()

# Copy shapefiles to a file geodatabase
for fc in featureclasses:
```

This sample is kind of fancy – lets just pull out what we need

… and just print out the feature classes for now

```
import arcpy

# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"

# Use the ListFeatureClasses function to return a list of
#  shapefiles.
featureclasses = arcpy.ListFeatureClasses()

# Copy shapefiles to a file geodatabase
for fc in featureclasses:
    print (fc)
```

This sample is kind of fancy – lets just pull out what we need

… also update the comments to keep them relevant to what we are doing

```
import arcpy

# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"

# Use the ListFeatureClasses function to return a list of
#  shapefiles.
featureclasses = arcpy.ListFeatureClasses()

# Copy shapefiles to a file geodatabase
for fc in featureclasses:
    print (fc)
```

This sample is kind of fancy – lets just pull out what we need

… also update the comments to keep them relevant to what we are doing

```
import arcpy

# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"

# Use the ListFeatureClasses function to return a list of feature classes.

featureclasses = arcpy.ListFeatureClasses()

# Print out each feature class name
for fc in featureclasses:
    print (fc)
```

We must set the workspace environment

```
import arcpy

# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"

# Use the ListFeatureClasses function to return a list of feature classes.

featureclasses = arcpy.ListFeatureClasses()

# Print out each feature class name
for fc in featureclasses:
    print (fc)
```

We must set the workspace environment

That is how LstFeatureClasses knows what workspace to list out

```
import arcpy

# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"

# Use the ListFeatureClasses function to return a list of feature classes

featureclasses = arcpy.ListFeatureClasses()

# Print out each feature class name
for fc in featureclasses:
    print (fc)
```

Lets remove some blank lines

```
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
```

Lets remove some blank lines

```
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
```

…and merge in our existing script

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
if arcpy.Exists(output):
    arcpy.Delete_management(output)
arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
```

…and merge in our existing script

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
if arcpy.Exists(output):
    arcpy.Delete_management(output)
arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
```

We will comment out the lines that run the Buffer tool for now

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
# if arcpy.Exists(output):
#     arcpy.Delete_management(output)
# arcpy.Buffer_analysis(input, output, "100 Feet")
# print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
```

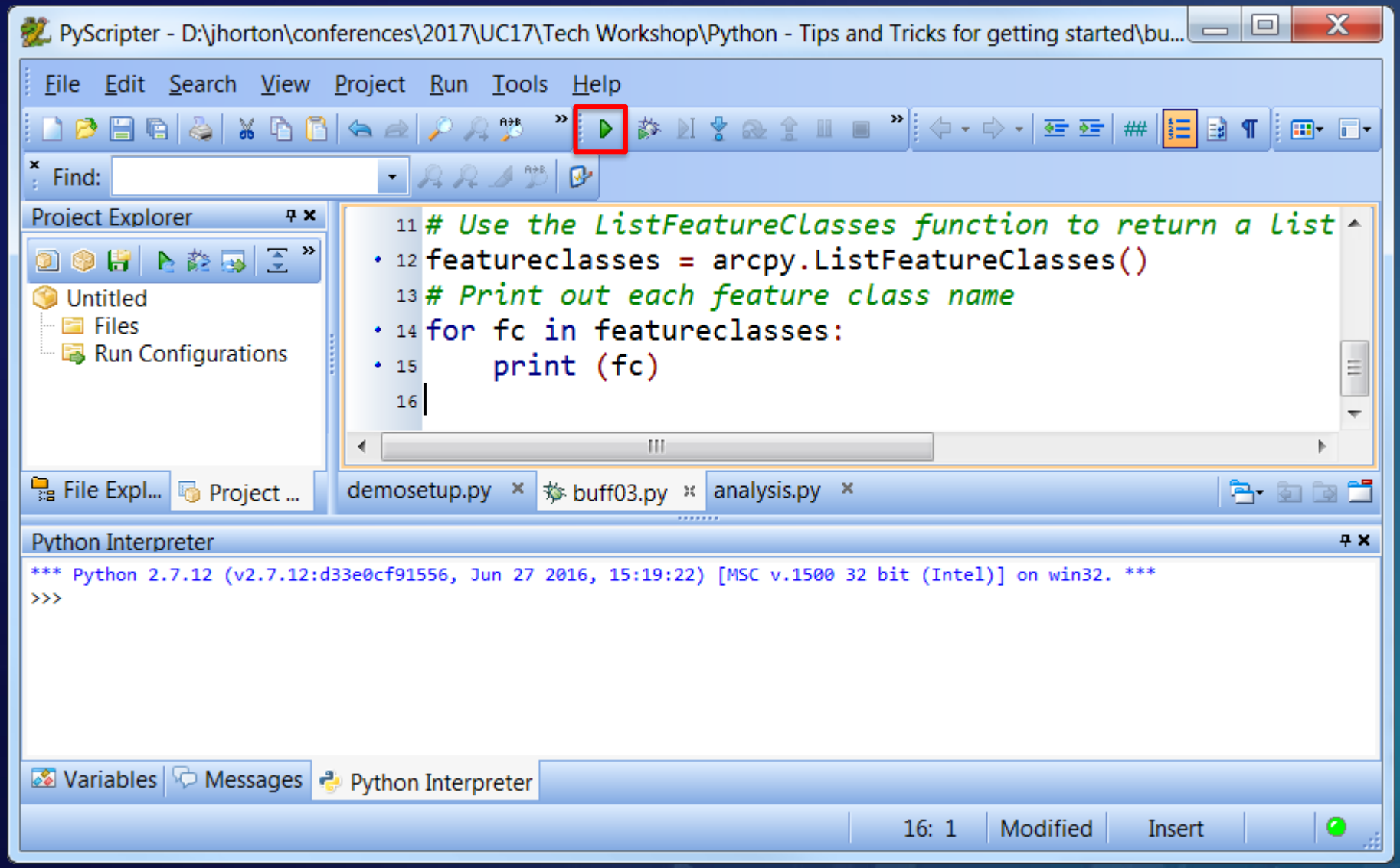
This script is still trying to use the data from the sample

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
# if arcpy.Exists(output):
#     arcpy.Delete_management(output)
# arcpy.Buffer_analysis(input, output, "100 Feet")
# print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = "c:/base"
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
```

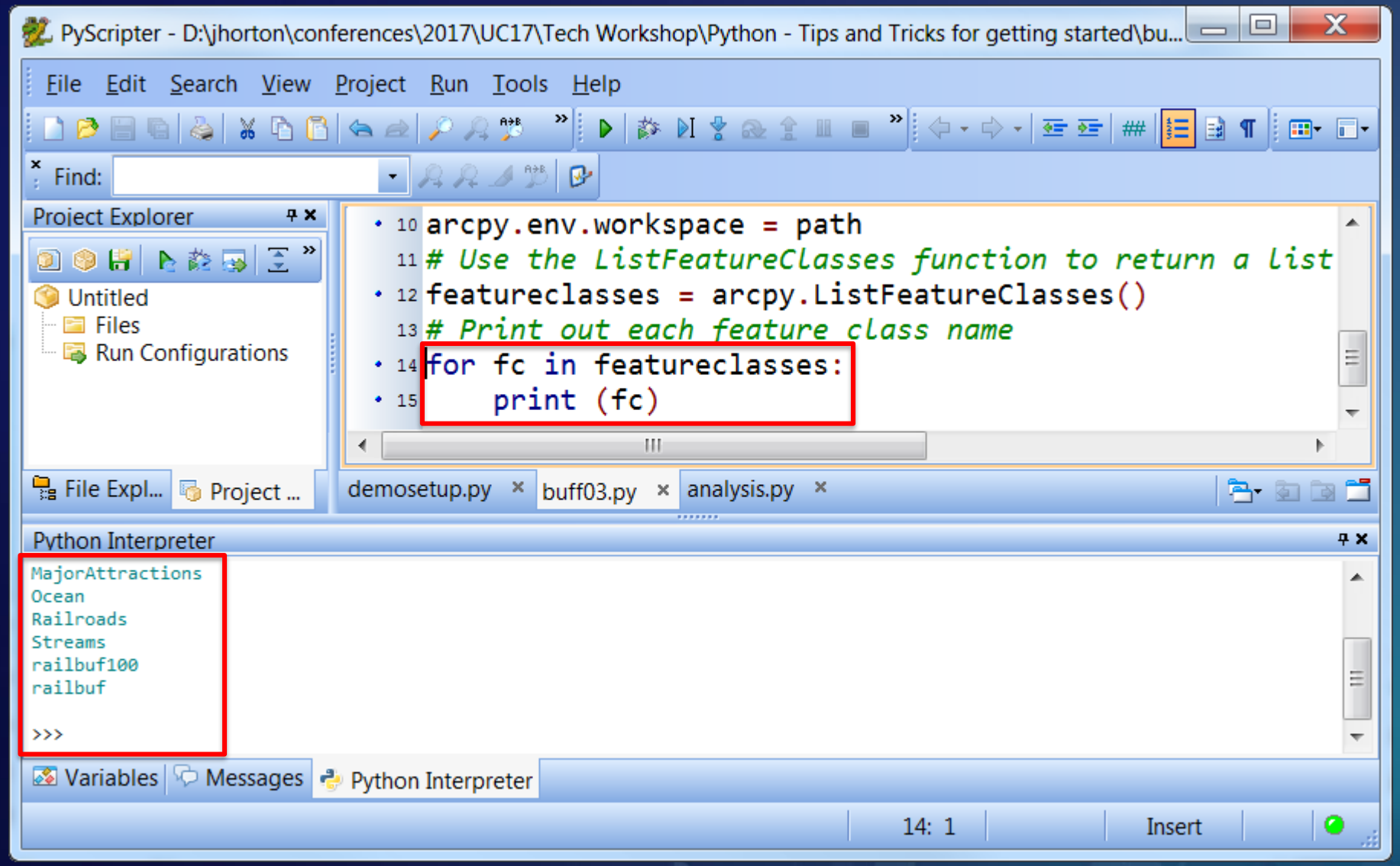
Substitute our path variable for the workspace

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
# if arcpy.Exists(output):
#     arcpy.Delete_management(output)
# arcpy.Buffer_analysis(input, output, "100 Feet")
# print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
```

Lets test what we have so far



It prints out all of the feature classes



Now that we got this far, lets start setting it up to run the Buffer tool

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
# if arcpy.Exists(output):
#     arcpy.Delete_management(output)
# arcpy.Buffer_analysis(input, output, "100 Feet")
# print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
```

The input and output variables are currently hard-coded to our test data

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
# if arcpy.Exists(output):
#     arcpy.Delete_management(output)
# arcpy.Buffer_analysis(input, output, "100 Feet")
# print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
```

Set up the input and output variables to use each feature class in the loop

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
input = path + "/railroads"
output = path + "/railbuf"
# if arcpy.Exists(output):
#     arcpy.Delete_management(output)
# arcpy.Buffer_analysis(input, output, "100 Feet")
# print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
```

… and remove the original hard-coded input and output variables

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"



# if arcpy.Exists(output):
#     arcpy.Delete_management(output)
# arcpy.Buffer_analysis(input, output, "100 Feet")
# print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
```

… and remove the original hard-coded input and output variables

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# if arcpy.Exists(output):
#     arcpy.Delete_management(output)
# arcpy.Buffer_analysis(input, output, "100 Feet")
# print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
```

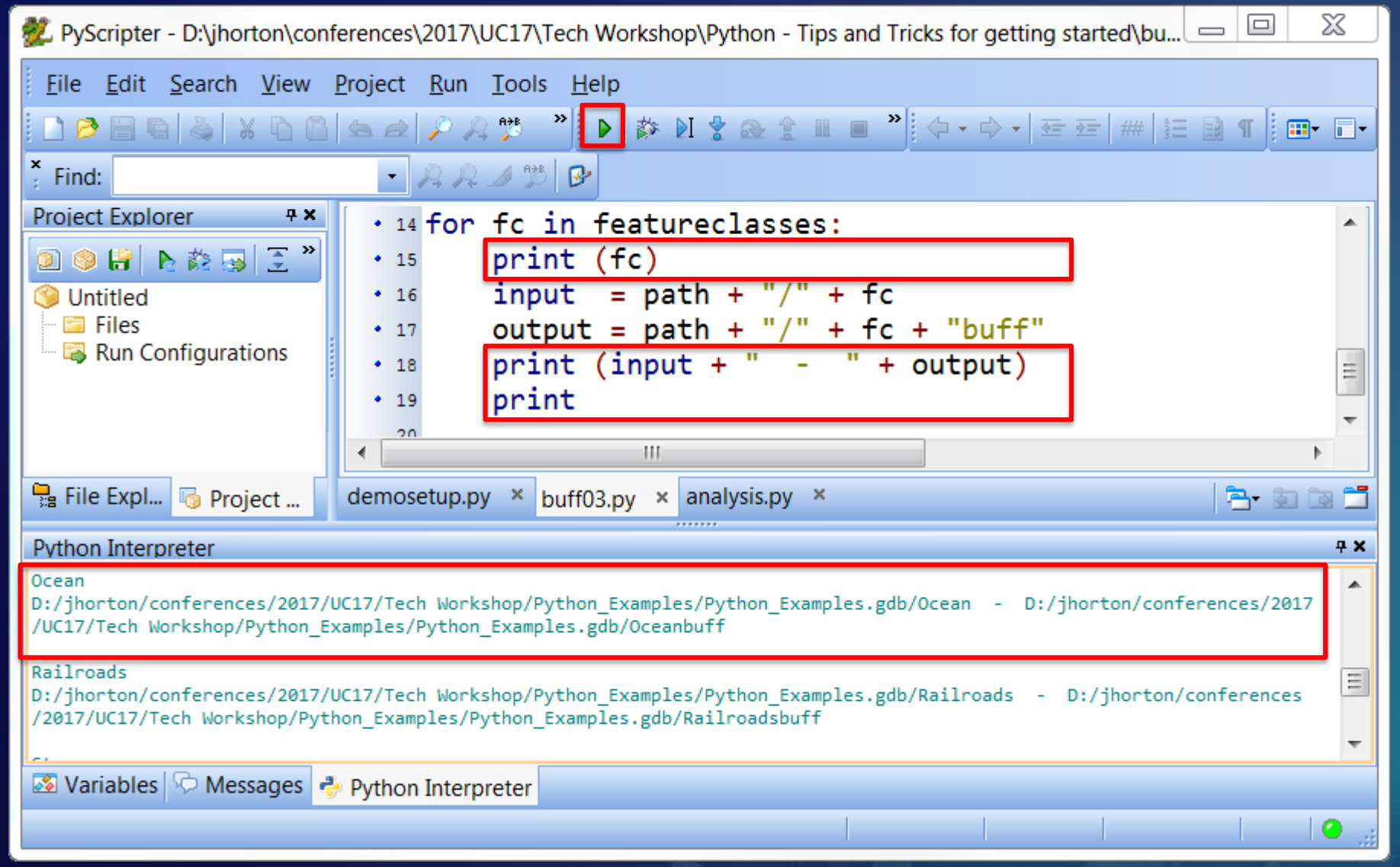
… and remove the original hard-coded input and output variables

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# if arcpy.Exists(output):
#     arcpy.Delete_management(output)
# arcpy.Buffer_analysis(input, output, "100 Feet")
# print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
```

Print out our new variable values

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# if arcpy.Exists(output):
#     arcpy.Delete_management(output)
# arcpy.Buffer_analysis(input, output, "100 Feet")
# print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    print (input + "  -  " + output)
    print ()
```

Lets confirm that it correctly sets our input and output variables



Now alter the print statements and run the buffer

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# if arcpy.Exists(output):
#     arcpy.Delete_management(output)
# arcpy.Buffer_analysis(input, output, "100 Feet")
# print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    print (input + "  -  " + output)
    print ()
```

Get ready to copy and paste the buffer statements into the for loop

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# if arcpy.Exists(output):
#     arcpy.Delete_management(output)
# arcpy.Buffer_analysis(input, output, "100 Feet")
# print ("done")
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    print (input + "  -  " + output)
    print ()
```

Get ready to copy and paste the buffer statements into the for loop

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"



# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    print (input + "  -  " + output)
    print ()
```

Get ready to copy and paste the buffer statements into the for loop

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    print (input + "  -  " + output)
    print ()
```

Remove the print() statements we used for testing

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    print (input + "  -  " + output)
    print ()
```

Remove the print() statements we used for testing

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
```

Alter the remaining print statement

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print (fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
```

Alter the remaining print statement

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Print out each feature class name
for fc in featureclasses:
    print ("Buffering " + fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
```

Paste in the lines that run the buffer tool

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class
for fc in featureclasses:
    print ("Buffering " + fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
# if arcpy.Exists(output):
#     arcpy.Delete_management(output)
# arcpy.Buffer_analysis(input, output, "100 Feet")
# print ("done")
```

Remove the comment characters (#) to activate these lines

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class
for fc in featureclasses:
    print ("Buffering " + fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
if arcpy.Exists(output):
    arcpy.Delete_management(output)
arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

And indent them properly so they run inside the for loop

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class
for fc in featureclasses:
    print ("Buffering " + fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
    arcpy.Buffer_analysis(input, output, "100 Feet")
    print ("done")
```

Un-indent the final print() statement, so it only prints “done” at the end

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class
for fc in featureclasses:
    print ("Buffering " + fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
    arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Challenge: How do you keep it from buffering a buffered feature class?

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class
for fc in featureclasses:
    print ("Buffering " + fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
    arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Challenge: How do you keep it from buffering a buffered feature class?

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class
for fc in featureclasses:
    print ("Buffering " + fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
    arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Challenge: How do you keep it from buffering a buffered feature class?

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in "buff"
for fc in featureclasses:
    print ("Buffering " + fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
    arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Challenge: How do you keep it from buffering a buffered feature class?

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in "buff"
for fc in featureclasses:
    print ("Buffering " + fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
    arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Answer: Don’t run the buffer if the input ends in “buff”

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in “buff”
for fc in featureclasses:
    print ("Buffering " + fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff“
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
     if input[-4:] <> "buff":
        arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

The print() is now misleading – we are not buffering every feature class

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in "buff"
for fc in featureclasses:
    print ("Buffering " + fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
     if input[-4:] <> "buff":
        arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

The print() is now misleading – we are not buffering every feature class

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in "buff"
for fc in featureclasses:
    print ("Buffering " + fc)
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
     if input[-4:] <> "buff":

        arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Put it inside the if so it only prints if we actually buffer the feature class

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in "buff"
for fc in featureclasses:

    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
     if input[-4:] <> "buff":
        print ("Buffering " + fc)
        arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Put it inside the if so it only prints if we actually buffer the feature class

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in “buff”
for fc in featureclasses:

    input  = path + "/" + fc
    output = path + "/" + fc + "buff“
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
     if input[-4:] <> "buff":
        print ("Buffering " + fc)
        arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

Put it inside the if so it only prints if we actually buffer the feature class

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in "buff"
for fc in featureclasses:
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
     if input[-4:] <> "buff":
        print ("Buffering " + fc)
        arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

How did [-4:] specify the last four characters?

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in "buff"
for fc in featureclasses:
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
    if input[-4:] <> "buff":
        print ("Buffering " + fc)
        arcpy.Buffer_analysis(input, output, "100 Feet")
print ("done")
```

How did [-4:] specify the last four characters?

if input[-4:] <> "buff“:

basically says:

if the last 4 characters of input are not “buff”

How did [-4:] specify the last four characters?

if input[-4:] <> "buff“:

basically says:

if the last 4 characters of input are not “buff”

Strings are indexed between the letters like this:

How did [-4:] specify the last four characters?

if input[-4:] <> "buff“:

basically says:

if the last 4 characters of input are not “buff”

Strings are indexed between the letters like this:

"abcdefg"[0:3] returns “abc”

"abcdefg"[:3] also returns “abc”

"abcdefg"[3:6] returns “def”

"abcdefg"[-4:-1] also returns “def”

"abcdefg"[3:7] returns “defg”

"abcdefg"[3:] also returns “defg”

"abcdefg"[-4:7] returns “defg”

"abcdefg"[-4:] also returns “defg”

How did [-4:] specify the last four characters?

if input[-4:] <> "buff“:

basically says:

if the last 4 characters of input are not “buff”

Strings are indexed between the letters like this:

"abcdefg"[0:3] returns “abc”

"abcdefg"[:3] also returns “abc”

"abcdefg"[3:6] returns “def”

"abcdefg"[-4:-1] also returns “def”

"abcdefg"[3:7] returns “defg”

"abcdefg"[3:] also returns “defg”

"abcdefg"[-4:7] returns “defg”

"abcdefg"[-4:] also returns “defg”

How did [-4:] specify the last four characters?

if input[-4:] <> "buff“:

basically says:

if the last 4 characters of input are not “buff”

Strings are indexed between the letters like this:

"abcdefg"[0:3] returns “abc”

"abcdefg"[:3] also returns “abc”

"abcdefg"[3:6] returns “def”

"abcdefg"[-4:-1] also returns “def”

"abcdefg"[3:7] returns “defg”

"abcdefg"[3:] also returns “defg”

"abcdefg"[-4:7] returns “defg”

"abcdefg"[-4:] also returns “defg”

How did [-4:] specify the last four characters?

if input[-4:] <> "buff“:

basically says:

if the last 4 characters of input are not “buff”

Strings are indexed between the letters like this:

"abcdefg"[0:3] returns “abc”

"abcdefg"[:3] also returns “abc”

"abcdefg"[3:6] returns “def”

"abcdefg"[-4:-1] also returns “def”

"abcdefg"[3:7] returns “defg”

"abcdefg"[3:] also returns “defg”

"abcdefg"[-4:7] returns “defg”

"abcdefg"[-4:] also returns “defg”

How did [-4:] specify the last four characters?

if input[-4:] <> "buff“:

basically says:

if the last 4 characters of input are not “buff”

Strings are indexed between the letters like this:

"abcdefg"[0:3] returns “abc”

"abcdefg"[:3] also returns “abc”

"abcdefg"[3:6] returns “def”

"abcdefg"[-4:-1] also returns “def”

"abcdefg"[3:7] returns “defg”

"abcdefg"[3:] also returns “defg”

"abcdefg"[-4:7] returns “defg”

"abcdefg"[-4:] also returns “defg”

How did [-4:] specify the last four characters?

if input[-4:] <> "buff“:

basically says:

if the last 4 characters of input are not “buff”

Strings are indexed between the letters like this:

"abcdefg"[0:3] returns “abc”

"abcdefg"[:3] also returns “abc”

"abcdefg"[3:6] returns “def”

"abcdefg"[-4:-1] also returns “def”

"abcdefg"[3:7] returns “defg”

"abcdefg"[3:] also returns “defg”

"abcdefg"[-4:7] returns “defg”

"abcdefg"[-4:] also returns “defg”

How did [-4:] specify the last four characters?

if input[-4:] <> "buff“:

basically says:

if the last 4 characters of input are not “buff”

Strings are indexed between the letters like this:

"abcdefg"[0:3] returns “abc”

"abcdefg"[:3] also returns “abc”

"abcdefg"[3:6] returns “def”

"abcdefg"[-4:-1] also returns “def”

"abcdefg"[3:7] returns “defg”

"abcdefg"[3:] also returns “defg”

"abcdefg"[-4:7] returns “defg”

"abcdefg"[-4:] also returns “defg”

How did [-4:] specify the last four characters?

if input[-4:] <> "buff“:

basically says:

if the last 4 characters of input are not “buff”

Strings are indexed between the letters like this:

"abcdefg"[0:3] returns “abc”

"abcdefg"[:3] also returns “abc”

"abcdefg"[3:6] returns “def”

"abcdefg"[-4:-1] also returns “def”

"abcdefg"[3:7] returns “defg”

"abcdefg"[3:] also returns “defg”

"abcdefg"[-4:7] returns “defg”

"abcdefg"[-4:] also returns “defg”

How did [-4:] specify the last four characters?

if input[-4:] <> "buff“:

basically says:

if the last 4 characters of input are not “buff”

Strings are indexed between the letters like this:

"abcdefg"[0:3] returns “abc”

"abcdefg"[:3] also returns “abc”

"abcdefg"[3:6] returns “def”

"abcdefg"[-4:-1] also returns “def”

"abcdefg"[3:7] returns “defg”

"abcdefg"[3:] also returns “defg”

"abcdefg"[-4:7] returns “defg”

"abcdefg"[-4:] also returns “defg”

How did [-4:] specify the last four characters?

if input[-4:] <> "buff“:

basically says:

if the last 4 characters of input are not “buff”

Strings are indexed between the letters like this:

"abcdefg"[0:3] returns “abc”

"abcdefg"[:3] also returns “abc”

"abcdefg"[3:6] returns “def”

"abcdefg"[-4:-1] also returns “def”

"abcdefg"[3:7] returns “defg”

"abcdefg"[3:] also returns “defg”

"abcdefg"[-4:7] returns “defg”

"abcdefg"[-4:] also returns “defg”

That is how we use if to only run buffer if the last 4 characters are not “buff”

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in "buff"
for fc in featureclasses:
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
    if input[-4:] <> "buff":
        print ("Buffering " + fc)
        arcpy.Buffer_analysis(input, output, "100 Feet")
print ("Done")
```

Almost done…

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in "buff"
for fc in featureclasses:
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
    if input[-4:] <> "buff":
        print ("Buffering " + fc)
        arcpy.Buffer_analysis(input, output, "100 Feet")
print ("Done")
```

Almost done…

```
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in "buff"
for fc in featureclasses:
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
    if input[-4:] <> "buff":
        print ("Buffering " + fc)
        arcpy.Buffer_analysis(input, output, "100 Feet")
print ("Done")
```

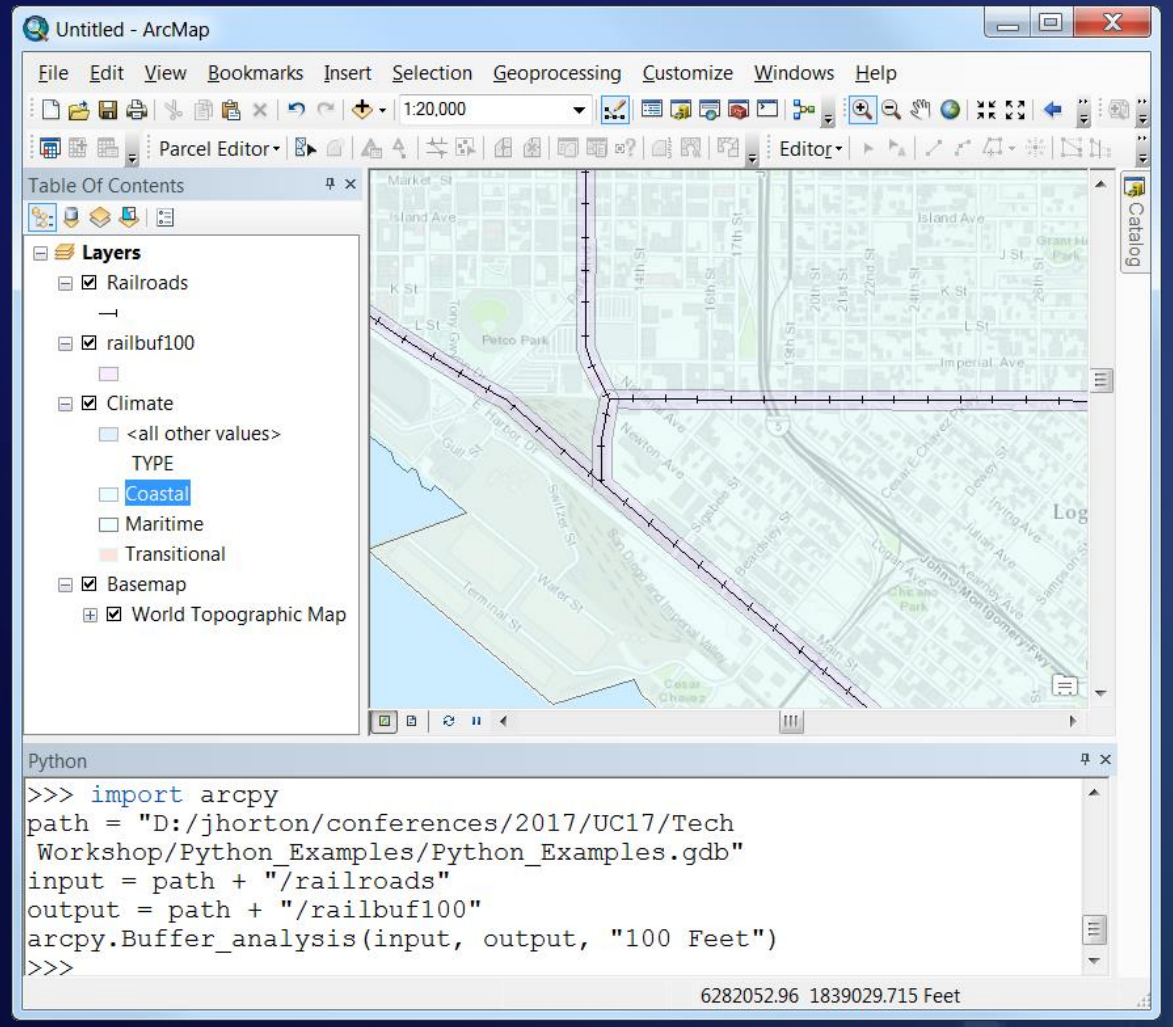
Add some header comments

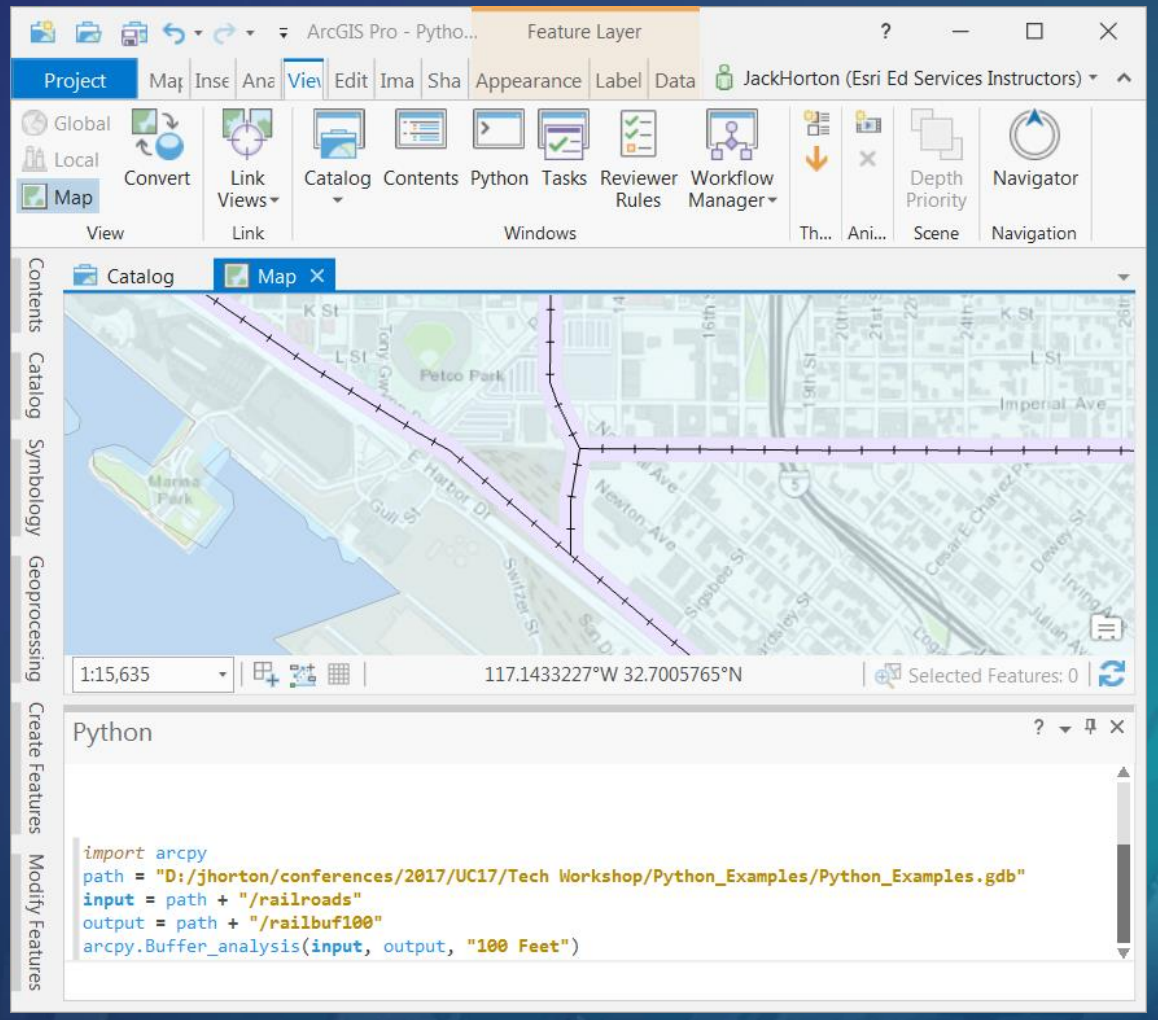
```
# BufferAll.py
# Jack Horton   June 30, 2018
# Buffers all the feature classes in a workspace, adding “buff” to each output
#
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in "buff"
for fc in featureclasses:
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
    if input[-4:] <> "buff":
        print ("Buffering " + fc)
        arcpy.Buffer_analysis(input, output, "100 Feet")
print ("Done")
```

Our completed script!

```
# BufferAll.py
# Jack Horton   June 30, 2018
# Buffers all the feature classes in a workspace, adding “buff” to each output
#
import arcpy
path = "D:/jhorton/conferences/2018/UC18/TechWorkshop/Python_Examples/Python_Examples.gdb"
# Set the workspace for ListFeatureClasses
arcpy.env.workspace = path
# Use the ListFeatureClasses function to return a list of feature classes
featureclasses = arcpy.ListFeatureClasses()
# Buffer each feature class that does not end in "buff"
for fc in featureclasses:
    input  = path + "/" + fc
    output = path + "/" + fc + "buff"
    if arcpy.Exists(output):
        arcpy.Delete_management(output)
    if input[-4:] <> "buff":
        print ("Buffering " + fc)
        arcpy.Buffer_analysis(input, output, "100 Feet")
print ("Done")
```

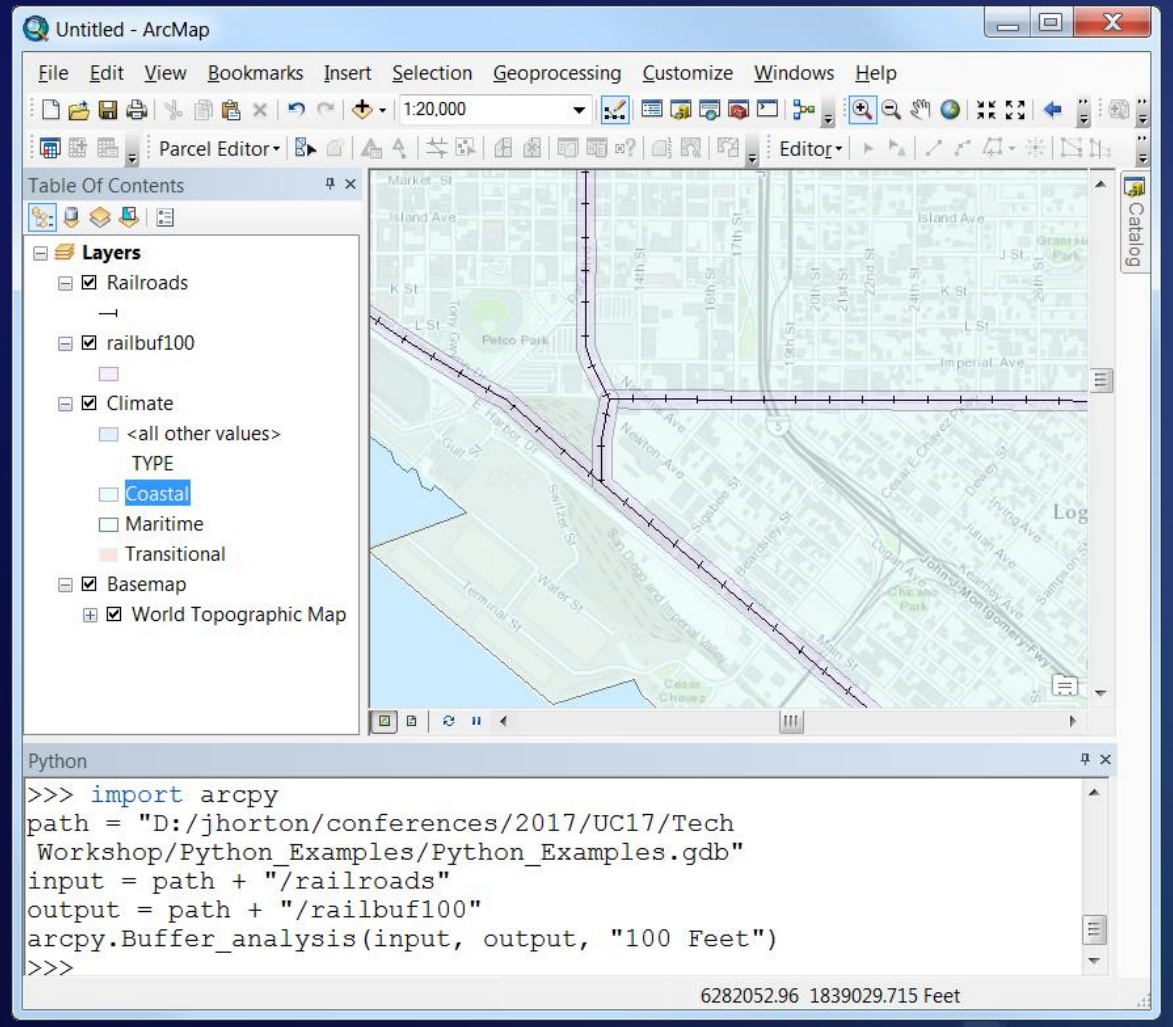
The same script runs in ArcMap or ArcGIS Pro

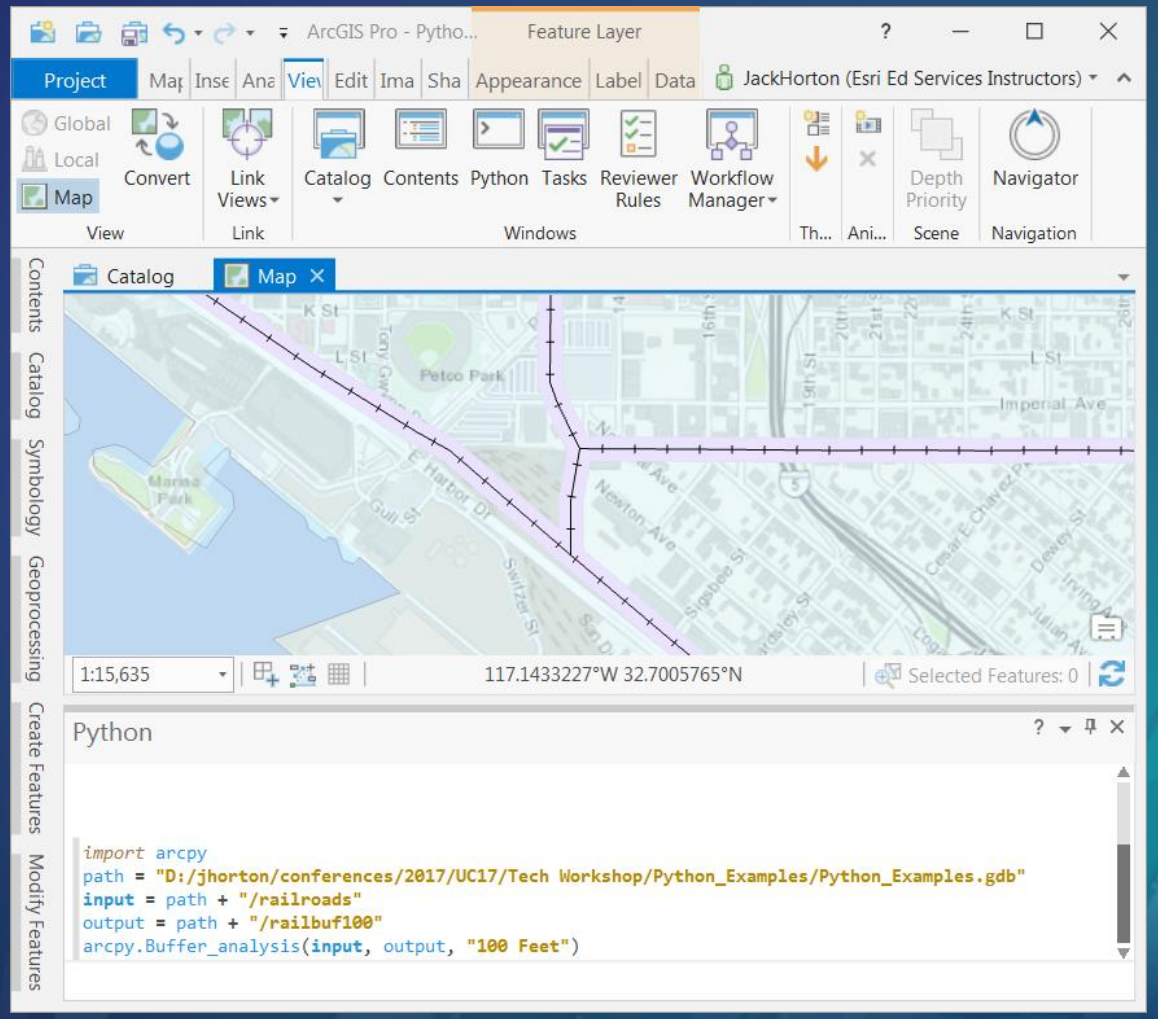




The same script runs in ArcMap or ArcGIS Pro

*** scripts that manipulate the user interface will, of course, be different**





Top 5 Python tips and tricks

Top 5 Python tips and tricks

1. **Start with sample code**

   from the help system or an internet search

Top 5 Python tips and tricks

1. Start with sample code
   from the help system or an internet search
2. Use meaningful names for your variables
   and keep it simple

Top 5 Python tips and tricks

1. Start with sample code

   from the help system or an internet search
2. Use meaningful names for your variables

   and keep it simple
3. Python is CasE SeNsITivE

   so use copy/paste to reduce spelling errors

Top 5 Python tips and tricks

1. Start with sample code

   from the help system or an internet search
2. Use meaningful names for your variables

   and keep it simple
3. Python is CasE SeNsITivE

   so use copy/paste to reduce spelling errors
4. Put parentheses after function calls, even if they take no parameters

   Example: arcpy.ListFeatureClasses()

Top 5 Python tips and tricks

1. Start with sample code
   from the help system or an internet search
2. Use meaningful names for your variables
   and keep it simple
3. Python is CasE SeNsITivE
   so use copy/paste to reduce spelling errors
4. Put parentheses after function calls, even if they take no parameters
   Example: arcpy.ListFeatureClasses()
5. Use print() functions or a debugger to figure out what your script is really doing
   and test as you go

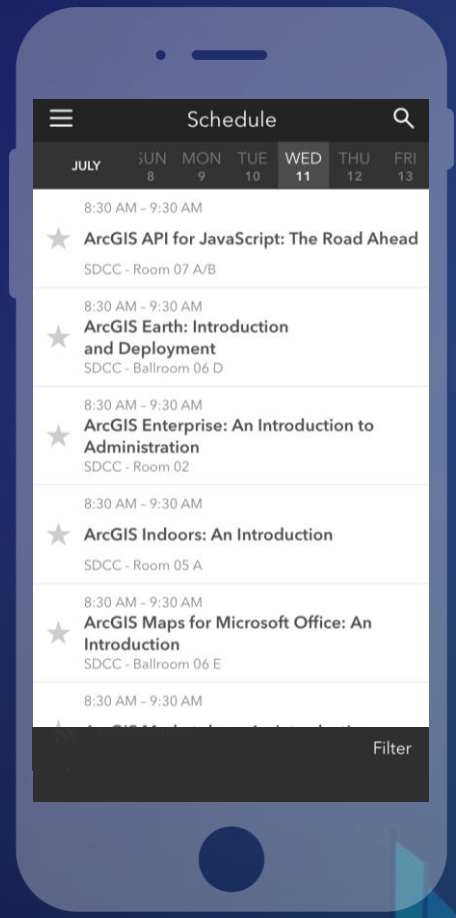
Top 5 Python tips and tricks

1. Start with sample code
   from the help system or an internet search
2. Use meaningful names for your variables
   and keep it simple
3. Python is CasE SeNsITivE
   so use copy/paste to reduce spelling errors
4. Put parentheses after function calls, even if they take no parameters
   Example: arcpy.ListFeatureClasses()
5. Use print() functions or a debugger to figure out what your script is really doing
   and test as you go

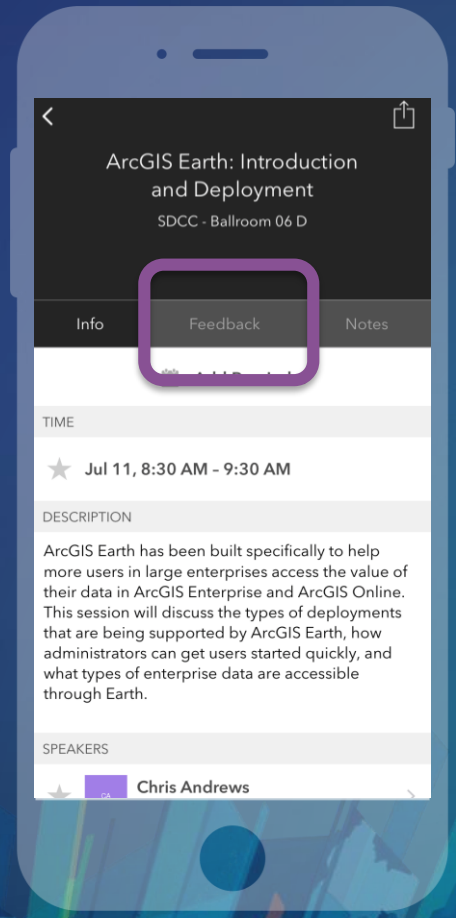
Please Take Our Survey on the App

Download the Esri Events app and find your event

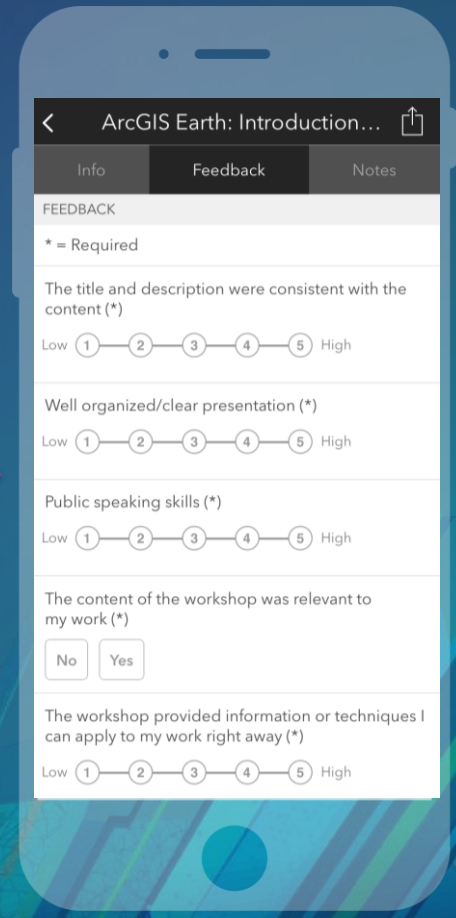
Select the session you attended



Scroll down to find the feedback section



Complete answers and select “Submit”




esri
THE
SCIENCE
OF
WHERE