Click here to watch the Lesson 5 video

Error handling

Chapter PSA book, 7

What to anticipate and how to handle errors

NR426

Programming for GIS I

Lesson 5

Instructor:

Elizabeth Tulanowski | ESS Department

What does Lesson 5 cover?

Lesson 5 is one class session

Identifying common errors in Python

Handling errors in Python

Improving error messages and user feedback

Running user friendly scripts

Quote of the day

"If debugging is the process of removing software bugs, then programming must be the process of putting them in."

- Edsger W. Dijkstra

Identifying common errors and exceptions

Refer to PSA book 7.2 and 7.11

'Typoes' , case sensitivity	Variables not defined
Missing data, data doesn't exist	Incorrect tool syntax
Syntax issues: : indentation ( )	Overwrite output not on
Module not imported	== vs. = for conditionals

1. Proofread your script!
   - Look for cues from the IDE, too
2. Use print messages throughout to see what lines of code have been executed
3. Comment out lines of code that don't work so you can fix them later

Tips for finding and avoiding errors

1. Proofread your script!
2. Use print messages throughout to see what lines of code have been executed
3. Comment out lines of code that don't work so you can fix them later
4. Add checks for data existence and type
5. Make use of the Debugging tools (PSA Book, 7.3 – p. 228)
6. Use try, traceback

The try-except block

- One of a few ways you can anticipate and handle errors in a script.
  - *Try to do something, and if anything goes wrong, raise an exception.*
- Allows the script to continue running, and think it executed successfully, even if a step failed.
  - Without the try statement, the script would just abruptly end when it encounters the error.

```
try:
        arcpy.Buffer_analysis (input, output, "2 Miles")
except:
        print arcpy.GetMessages()
```

Refer to PSA Book, 7.8 for more try-except examples

The try-except block

Refer to PSA Book 7.8 – 7.9

- Raise exceptions for specific, anticipated errors

```
x = input("First integer: ")
y = input("Second integer: ")
print(int(x)/int(y))
```

```
ZeroDivisionError: division by zero
```

```
try:
    x = input("First integer: ")
    y = input("Second integer: ")
    print(int(x)/int(y))
except ZeroDivisionError:
    print("The second integer cannot be zero.")
```

or

```
try:
    x = input("First integer: ")
    y = input("Second integer: ")
    print(int(x)/int(y))
except ZeroDivisionError:
    print("The second integer cannot be zero.")
except ValueError:
    print("Only integers are valid entries.")
```

- Anticipate and trap arcpy errors

```
import arcpy
arcpy.env.workspace = "C:/Data"
in_features = "streams.shp"
out_features = "streams.shp"
try:
    arcpy.CopyFeatures_management(in_features, out_features)
except arcpy.ExecuteError:
    print(arcpy.GetMessages(2))
except:
    print("There has been a nontool error.")
```

With and without error handling

```
import arcpy
arcpy.env.workspace = "C:/Student/CityOfSanAntonio.gdb"
try:
    #Create a query statement to find records where Day of Week is Frid
    qry = "DOW = 'Fri' Or DOW = 'Sat'"

    # First create the in-memory layer from the feature class
    flayer = arcpy.MakeFeatureLayer_management("Burglary","Burglary_Lay

    # Next perform the selection
    arcpy.SelectLayerByAttribute_management(flayer, "New_Selection", qry)

    #Get and report the number of selected features
    cnt = arcpy.GetCount_management(flayer)
    print("The number of selected records is: " + str(cnt))

# This is a technique for better-formatted error messag
except Exception as e:
    print("Error: " + e.args[0])
```

Lesson 4 Sample_SelectLayerAttribute

```
"C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe" "D:/NR426-4
Error: Failed to execute. Parameters are not valid.
ERROR 000732: Input Features: Dataset Burglry does not exist or is not supported
Failed to execute (MakeFeatureLayer).


Process finished with exit code 0
```

With error handling

Lesson 4 Sample_SelectLayerAttribute no try

```
"C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe" "D:/NR426-427/NR426 Materials/Lesson 4 - Working with tables/Lesson
Traceback (most recent call last):
  File "D:/NR426-427/NR426 Materials/Lesson 4 - Working with tables/Lesson 4 Sample_SelectLayerAttribute no try.py", line 13, in <module>
    flayer = arcpy.MakeFeatureLayer_management("Burglry","Burglary_Layer")
  File "C:\Program Files\ArcGIS\Pro\Resources\ArcPy\arcpy\management.py", line 7682, in MakeFeatureLayer
    raise e
  File "C:\Program Files\ArcGIS\Pro\Resources\ArcPy\arcpy\management.py", line 7679, in MakeFeatureLayer
    retval = convertArcObjectToPythonObject(gp.MakeFeatureLayer_management(*gp_fixargs((in_features, out_layer, where_clause, workspace, fi
  File "C:\Program Files\ArcGIS\Pro\Resources\ArcPy\arcpy\geoprocessing\_base.py", line 511, in <lambda>
    return lambda *args: val(*gp_fixargs(args, True))
arcgisscripting.ExecuteError: Failed to execute. Parameters are not valid.
ERROR 000732: Input Features: Dataset Burglry does not exist or is not supported
Failed to execute (MakeFeatureLayer).


Process finished with exit code 1
```

Without error handling

In-class hands-on activities

1. Find the errors (on screen)
2. Work together to identify places to add error handling
   - Check if input data exists
   - Check if input is the right type of data (the code needs a vector input)
   - Make sure that the output data doesn't exist (or turn on overwrite)
   - Make use of Try-except
   - Add print statements
   - Improve readability (comments, blank lines, etc…)

Submit your revised code to Canvas as Lab 5

Tips for success: testing your script

- Make sure your script runs successfully with *hard-coded* data before making it dynamic/allowing user arguments
- Build complexity in your scripts
  - Run it on a single dataset before running a list of datasets in a loop
  - Run one tool on a list of data before running a whole series of tools
  - Run a loop without the tool active, just outputting a sample string to see if your loops and logic are correct. Then make the tool run.
  - Run scripts on small datasets/subsets before running on that 1gb raster. You'll have to wait 10 minutes to see if it would work!
- It is also a good idea to test the script on data you know WON'T work, so that you can make sure your error handling is working properly.
  - Try your best to break that script and see how it handles. Are the messages useful?

Refer to PSA Book 7.5, p. 233 for more ideas

What did you just learn?

- Provide examples of some of the most common syntax errors
- Provide examples of some of the most common data-related errors
- Name and describe 3 techniques used for avoiding errors

Lab 5

In-class activity

- Find the errors
- Identify ways to add error checking

Take existing code and improve it to include some error checking techniques identified above, readability and messaging