**🗓️ Class Notes – Week 10 (March 25, 2025)**

**Course:** Programming for GIS (GISC 2335)  
**Professor:** Mr. Erik Bushland  
**Meeting Type:** Lecture + Lab  
**Topic:** Chapter 7 – *Debugging and Error Handling*

**🧠 Key Concepts from Class**

**📘 Chapter Focus:**

* Understanding how to identify, catch, and fix errors in Python code related to GIS workflows.
* Emphasis on **debugging tools**, **error types**, and **best practices** for writing resilient code.

**🔍 Topics Covered:**

1. **Syntax Errors** vs. **Runtime Errors** vs. **Logical Errors**
2. **Using try/except blocks** in Python to catch errors without crashing the script.
   * Example: catching ValueError, KeyError, and TypeError.
3. **Debugging in ArcGIS Pro**
   * Practical demos on how to use ArcGIS Pro’s built-in debugger.
4. **Logging and Print Statements**
   * When to use logging vs. print()
   * Using logging module for long-term debugging and tracking issues.
5. **Common GIS-related Python Errors**
   * File path issues (e.g., incorrect slashes)
   * Missing spatial references
   * Failing geoprocessing tools due to incorrect input parameters

**🛠️ Lab Activity:**

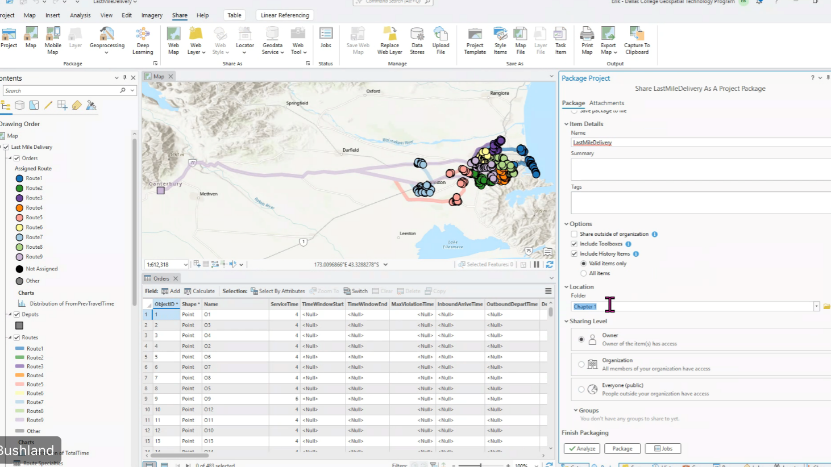
* Students worked on debugging pre-written Python scripts for:
  + Reclassifying raster data
  + Buffering feature classes
* Emphasis on interpreting traceback messages and finding the root of the problem.

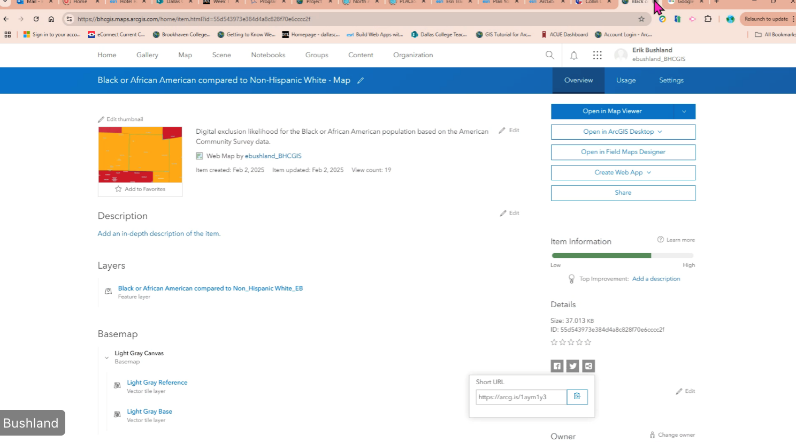
**📝 Assignment Due**

* **Ch. 7 Exercise** – Due by **5:00 PM next Tuesday** (April 1, 2025)  
  Focus: Apply debugging techniques to faulty GIS scripts provided in the assignment.

**📸 Homework Reminder**

* This week’s assignment will start with broken code.
* Use **print statements**, **commenting**, and **debugging tools** to fix it.
* **Take screenshots** showing how you debugged it!





**✅ Midterm Follow-up**

* **Midterm grades were re-reviewed**: Some free-response answers were regraded manually.
  + Answers didn’t need to match Mr. Bushland’s phrasing exactly.
  + Examples: “hashtag” vs. “pound sign”; either were accepted with context.
* **Check your grades and comments**:
  + Some files were ungraded due to wrong submissions (e.g., wrong chapter or file type).
  + You must upload the **correct Python script or share project via ArcGIS Online**.

**🚨 Grading & Participation Reminders**

* If a grade is missing, check comments. You may have:
  + Submitted wrong file.
  + Forgotten to share project properly.
* You’re encouraged to **resubmit fixed work for regrading**.
* **Goal is learning, not failure** – low grades are an invitation to revise and learn.

**🔧 File Submission Instructions (ArcGIS Pro)**

**Correct way to submit your projects:**

1. In ArcGIS Pro: go to **Share → Share Project**.
2. On the right panel:
   * Name the package.
   * **Include toolboxes & history items**.
   * **Set sharing level to “Organization.”**
   * Choose a folder like Chapter7.
3. After packaging:
   * Go to **ArcGIS Online → My Content**.
   * Open project details.
   * Copy the **short URL** at the bottom-right and submit that link.

Do **not** submit .aprx files alone — the file paths break if your environment is different.

**🔧 File Submission Clarifications**

* **Do not submit .aprx files directly** – they often break due to pathing differences (e.g., your username on your C: drive is different from the instructor's).
* **Always share via ArcGIS Pro → Share → Package Project → Upload to Online → Share with Organization**.
* Once uploaded, go to **ArcGIS Online → My Content → Details → Copy the short URL** to submit.

🔁 **Mr. Bushland emphasized** this again to reduce grading delays.

**🛠 Debugging Concepts (Chapter 7)**

* Types of errors:
  + **Syntax Errors** – code that won’t compile or run.
  + **Runtime Errors** – code crashes during execution (e.g., missing file).
  + **Logical Errors** – code runs but gives wrong result.
* **Try/Except Blocks**: Use for trapping and handling exceptions.
* **Troubleshooting Process**:
  + Submit scripts with intentional errors (per assignment).
  + Instructor tests code by running it.
  + You get credit **even if it fails**, if failure matches expected behavior.

**🐞 Debugging Best Practices (Expanded)**

* **IDE settings**: Visual Studio (VS Code) and PyCharm allow:
  + Syntax checking with red squiggles
  + Hover tooltips for errors
  + Status bar notifications
* **Print statements after each geoprocessing step** help confirm whether a script reached that point.

💡 Mr. Bushland uses **print statements** and **commenting out entire sections** to isolate issues — especially in longer scripts.

**📊 ArcGIS Online Example**

* **Demo of ArcGIS Online “Black vs. Non-Hispanic White” racial disparities map.**
* Students are encouraged to explore Esri MOOCs (e.g., Climate Change course mentioned).

**🧠 Chapter 7: Debugging and Error Handling**

**🔹 Types of Errors in Python:**

| **Error Type** | **Description** |
| --- | --- |
| **Syntax Error** | Prevents the code from running. Caused by typos, punctuation, or bad structure. |
| **Exception** | Stops a running script mid-process (e.g., file not found, divide by zero). |
| **Logic Error** | Code runs but produces incorrect or undesirable results. |

**🔹 Example of a Syntax Error**

import arcpy

arcpy.env.workspace = "C:/Data/mydata.gdb"

fclist = arcpy.ListFeatureClasses()

for fc in fclist # ❌ missing colon

count = arcpy.GetCount\_management(fc)

print(count)

* ❗ **Fix**: Add colon after for fc in fclist:

**🖥️ IDE Tips & Visual Cues**

* **Common causes of syntax errors:**
  + Misspelled keywords
  + Missing colons/quotes
  + Inconsistent indentation (especially mixing tabs and spaces)
* **Copy-paste warning**: Don’t copy Python from Word, PowerPoint, or PDF without checking for:
  + Curly quotes → use straight quotes
  + Broken indentation or spacing
* **Red squiggly lines** = syntax hints
* IDEs like **Visual Studio** and **PyCharm** provide:
  + Pop-up error tooltips when hovering over squiggles
  + Line number indicators
  + Autocorrect/auto-indent commands

**🧵 Common Syntax Error Causes**

* Spelling, punctuation, and capitalization issues
* Missing colons, commas, or quotes
* Mixed tabs and spaces (especially when copying from Word/PDF)
* Curly quotes (“ ”) instead of straight quotes (" ")

📎 **Tip:** IDEs like **Visual Studio** and **PyCharm** show syntax issues with red squiggly lines. Hover your cursor for tooltips, and check the **status bar for error hints**.

**🔍 Section 7.4 – Using Debugging**

**Debugging** = a *methodical process* for finding and fixing errors in code.

🛠️ **Grace Hopper Origin Story**: The term “debugging” came from Admiral Grace Hopper, who literally pulled bugs out of computers in the 1950s!

* This origin reinforces the mindset that **debugging is normal and expected** — not a failure.

**🔧 Debugging Techniques:**

1. **Carefully read error messages**
2. **Add print statements** to test variable values
3. **Comment out sections** to isolate errors
4. **Use Python debuggers** (like built-in debuggers in VS Code or PyCharm)

**💬 Example: Error 000366**

arcpy.env.workspace = "C:/Data"

infcs = ["streams.shp", "floodzone.shp"]

outfc = "union.shp"

arcpy.Union\_analysis(infcs, outfc)

🟥 Error:

ExecuteError: Failed to execute. Parameters are not valid.

ERROR 000366: Invalid geometry type

🟨 **Explanation**: The Union tool only works with polygon feature classes. The script is trying to union a polyline (streams.shp) with a polygon.

🧠 **Tip:** Copy the error code (e.g., 000366) and search in **ArcGIS Pro Help** for detailed solutions.

**✅ Mr. Bushland’s Practical Tips**

* If your script doesn’t work, **comment out a suspected line** and re-run.
* If commenting that line allows the rest to work, the issue is in that line *or something it depends on*.
* Don’t be afraid to test parts of your code in isolation.
* **Goal is not to fail you** — the point is to learn through trial and correction.

**💡 Debugging in Practice**

**🖨️ Print Statements**

arcpy.Buffer\_analysis("roads.shp", "buffer.shp", "1000 METERS")

print("Buffer completed") # Helps confirm success

**🧹 Commenting Out Code**

# arcpy.Erase\_analysis("buffer.shp", "streams.shp", "erase.shp")

# arcpy.Clip\_analysis("erase.shp", "wetlands.shp", "clip.shp")

👉 Run one step at a time to isolate the error.

**🧑‍💻 Using a Debugger**

* Set breakpoints manually
* Run script in **debug mode**
* Track values of variables like fc, outfc, etc.
* In **PyCharm**, click between the **line number and code** to set breakpoints

**⚙️ Debugging Workflow Summary**

1. **Check for syntax errors**
2. **Set breakpoints** at key points
3. **Run the debugger**
4. **Step through** the code, watching variables update

**🧩 7.5 Debugging Tips & Tricks**

✅ Tips for real-world success:

* If ArcGIS Pro locks files, close it before running scripts.
* For big files, **test on smaller datasets**.
* Add **counters** in loops to reduce processing load.
* Use Ctrl+C to stop scripts manually.
* Use breakpoints where code repeats.

**🛑 7.6–7.7: Handling & Raising Exceptions**

* Use try/except blocks to handle errors gracefully:

try:

# code block

except:

print("Something went wrong.")

* You can raise custom exceptions too:

raise Exception("Invalid workspace")

**🔍 7.10 Other Error-Handling Methods**

Besides using try-except, you can:

* ✅ Validate field names with arcpy.ValidateFieldName()
* ✅ Validate table names with arcpy.ValidateTableName()
* ✅ Check for licenses: arcpy.CheckProduct() / arcpy.CheckExtension()
* ✅ Check for data locks (some tools won’t run if another app is using your input)

**👀 7.11 Watching for Common Errors**

**🔸 Common Python Code Errors**

* Misspelled words (e.g., arcpyy, os)
* Missing imports
* Case sensitivity (MyList ≠ mylist)
* Backslashes in paths (C:\Data\myfile.shp) → use **raw strings** (r"path") or double slashes
* Missing colons in for, if, while, etc.
* Indentation issues
* Confusing = (assignment) vs == (comparison)

**🔸 Common Geoprocessing Errors**

* Data doesn’t exist
* Forgot to enable overwrite: use arcpy.env.overwriteOutput = True
* File in use by ArcGIS Pro
* Forgot to use .getOutput() for results from tools like GetCount
* Tool output not compatible with tool input (e.g., polyline to Union which expects polygon)

**📝 Points to Remember**

* **Errors will happen** — it's expected in geoprocessing.
* **Debugging helps you find *where*, not always *why* the error happens.**
* Use multiple tools together:
  + Print statements
  + Commenting out
  + Breakpoints/debuggers
  + Checking for data/valid names
* Python scripts must follow **Python logic + ArcGIS geoprocessing rules**
* You can't anticipate every error — write scripts with graceful failure and flexibility.

**💬 Real-Life Debug Story (from Mr. Bushland)**

He once ran a script given by a supervisor that renamed every file in *every* folder — not just one subfolder as intended.

* The script technically "worked" but had logic errors.
* His own fix didn’t work because of a **simple typo**.
* But it proved the backup system worked 😅

“The issue wasn’t my code — it was the instructions. He assumed I knew what he knew.” — Mr. Bushland

**💬 Real-World Insight**

🗂️ **Debugging error with a real government script**:  
Mr. Bushland accidentally renamed **every file in a directory**, not just one folder, after following ambiguous instructions from his supervisor.

* The logic of the script was flawed.
* A simple **typo** caused his own fix to fail.
* The story illustrates:
  + The importance of clearly understanding scripts
  + How logic errors may not cause syntax errors but can still cause chaos

**✅ Final Tips for This Week**

* Debugging is **normal** and **expected** in GIS scripting.
* Use every tool at your disposal to catch, isolate, and fix bugs:
  + IDE features
  + Visual cues (squiggles)
  + ArcGIS-specific functions
* Screenshot your debugging process for your assignment!
* Be thorough — share your files online properly so instructors can test them.

**💡 Suggested Thoughtful Questions to Ask Professor Bushland**

1. **“What are your best practices for organizing code in a way that makes debugging easier?”**
2. **“Can you give real-world examples of debugging a GIS workflow that failed during automation?”**
3. **“How can we incorporate logging into large GIS projects for long-term monitoring and issue resolution?”**
4. **“Are there certain types of GIS data that tend to cause more scripting errors than others?”**
5. **“Do you recommend any plugins or external tools for debugging ArcPy scripts outside of ArcGIS Pro?”**
6. **“How can we best organize our Python scripts and toolboxes to reduce common debugging issues?”**
7. **“Can you show an example where error logging was critical in diagnosing a GIS automation failure?”**
8. **“Are there specific ArcPy tools or geoprocessing functions that are more error-prone?”**
9. **“When using shared projects on ArcGIS Online, what’s the best way to manage data updates or versioning?”**
10. **“If we wanted to build a public-facing dashboard, how would error handling change?”**
11. **“Is there a recommended linter or Python extension you'd suggest for ArcGIS users to catch errors before runtime?”**
12. **“Could you walk us through how to read a traceback message when debugging geoprocessing tools?”**
13. **“If logic errors aren't always flagged, how do you recommend testing for them systematically in GIS workflows?”**
14. **“When packaging a project with many toolboxes and datasets, is there a best practice to keep it lightweight?”**
15. **“Could you share an example of how you've used error handling to improve a real-world GIS automation task?”**
16. **What’s your go-to strategy for debugging logic errors when the script runs but gives wrong results?**
17. **Can you recommend beginner-friendly debugging tools or extensions for VS Code?**
18. **What are common exceptions in GIS workflows (e.g., with rasters, shapefiles, or projections)?**
19. **Are there GIS-specific linting tools to help catch errors before runtime?**
20. **How can we use try/except blocks effectively when working with geoprocessing tools in ArcPy?**
21. **“What’s your go-to process for isolating logic errors?”**
22. **“How can we make our Python scripts more resilient to user input errors?”**
23. **“Would you recommend adding try/except around every geoprocessing tool?”**
24. **“Can we use error logs instead of print statements in longer projects?”**
25. **“What should we include in screenshots to fully demonstrate our debugging process?”**

**💻 Platform Compatibility Tips**

* GIS Python scripts that work **locally** may break in **ArcGIS Online** due to:
  + Different file paths
  + Server structures
  + Missing geodatabases or different folder permissions
* Always test workflows **in the target environment**, especially when working across multiple systems (e.g., Mac vs. Windows).

**🧪 Version Control Advice**

* ArcGIS Online does **not** offer built-in version control for scripts.
* Recommendation: **save backup copies manually** and include:
  + Timestamps (e.g., bufferScript\_2025\_03\_25\_1330.py)
  + Clear versioning (e.g., v1, v2, final, draft)
* GitHub is a smart option for script versioning outside ArcGIS.

**💡 BONUS: Suggested Tools Mentioned**

* Visual Studio (VS Code) is **free with a student account** and supports:
  + Debugging
  + Syntax highlighting
  + Potential version control integration
* Open-source IDEs may offer even better debugging tools, depending on your workflow.