**GISC 2335 Class Notes - February 25, 2025**

**Week 6 – Chapter 5: Geoprocessing Using Python**

**Class Announcements & Assignments**

**Current Work:**

* We are now working on **Chapter 5: Geoprocessing Using Python**.
* **Chapter 4 exercise is due by midnight tonight** (February 25, 2025).
* You should also start working on **Exercise 5**, which is tied to Chapter 5.

**Important Reminders:**

* Always **check the exercise PDF first** to see if you need to download data.
* Example: **Chapter 4 did not require a data download**, but **Chapter 5 does**.
* **The midterm (after Spring Break)** will cover **Chapters 1-6**, so you do not need to work ahead past Chapter 6 until after the break.

**Future Plans:**

* After Spring Break and the midterm, the class may either:
  + Continue with the textbook, **OR**
  + Switch to an **applied Python project**.
* Either way, **Python scripting will continue to be a major focus**.

**Portfolio Project Update**

* **Portfolio projects** are part of the syllabus but will not be discussed in detail until after the midterm.
* Students who want to **start early** can but expect formal guidance after Spring Break.
* Depending on the class direction, the **portfolio project could tie into the applied project**.

**Submission Guidelines**

* **ArcGIS Pro Projects:**  
  Submit using the **Share** function in ArcGIS Pro to generate a **short URL**.  
  Do not submit zipped projects unless specifically instructed.
* **Python Scripts:**  
  Submit .py files directly.
* **Assignments Requiring Screenshots:**  
  Submit a **Word document with your screenshots**.

**Suggested Questions to Explore (For Career & Academic Growth)**

**Public Health & GIS Focus:**

1. What are the most important GIS tools for **epidemiological mapping**?
2. How can GIS environments and coordinate systems impact **disease surveillance data**?
3. What types of **Python automation** could streamline **public health reporting**?
4. How can I document my workflows for **reproducibility in public health research**?
5. What are the **licensing limitations** I might face working in a **public health agency**, and how can I work around them?

**Chapter 5 Key Topics and Notes**

**5.1 - 5.5: Introduction to ArcPy & Geoprocessing Tools**

* **ArcPy** allows Python to work directly with **ArcGIS Pro’s geoprocessing tools**.
* Start all scripts with:
* import arcpy
* Set the workspace (where inputs and outputs are stored):
* arcpy.env.workspace = "C:/GIS\_Project/Data"
* **Tool Syntax Example:**
* arcpy.Clip\_analysis("streams.shp", "study\_area.shp", "clipped\_streams.shp")
* Remember:
  + Python is **case-sensitive**.
  + Use forward slashes (/) or raw strings (r"") in file paths.
  + ArcPy provides access to tools, **but available tools depend on your license**.

**5.6: Working with Result Objects**

* **Result Objects:** Outputs returned by tools, which can be **reused in other tools**.
* Example:
* result = arcpy.Buffer\_analysis("roads.shp", "buffered\_roads.shp", "100 METERS")
* count = arcpy.GetCount\_management(result)
* This technique is essential for **chaining workflows** like you would in **ModelBuilder**.

**5.7: Working with Custom Toolboxes**

* **Custom toolboxes must be imported manually:**
* arcpy.ImportToolbox("path\_to\_toolbox.tbx", "alias")
* Licensing levels (Basic, Standard, Advanced) affect which tools you have access to.
* Be aware of **license differences between school and workplace environments**.

**5.8: Using ArcPy Functions**

* ArcPy has two types of functions:
  + **Tool Functions:** Directly linked to ArcGIS geoprocessing tools.
  + **Non-Tool Functions:** Helper functions like arcpy.Exists() and arcpy.ListFeatureClasses().
* Example:
* if arcpy.Exists("roads.shp"):
* print("File exists.")

**5.9 - 5.10: Classes & Coordinate Systems**

* **SpatialReference Class** simplifies handling coordinate systems.
* sr = arcpy.SpatialReference(2277)
* Instead of typing long coordinate system names, you can **use the Well-Known ID (WKID)**.
* Ensuring the correct **spatial reference** is critical when combining data from different sources.

**5.11: Using Environments**

* **Environments are global settings** that affect how tools run.
* Examples:
* arcpy.env.extent = "MAXOF"
* arcpy.env.overwriteOutput = True
* This helps maintain **consistent outputs**, especially when working with public datasets.

**5.12: Working with Tool Messages**

* Tools generate messages (info, warnings, errors) after running.
* You can retrieve these messages programmatically:
* print(arcpy.GetMessages())
* **Message Types:**
  + **Severity 0:** Info (everything is fine).
  + **Severity 1:** Warning (might work, but check results).
  + **Severity 2:** Error (tool failed to run).

**5.13: Working with Licenses**

* **ArcGIS Pro tools require appropriate licenses.**
* Example:
* print(arcpy.CheckProduct("arcinfo")) # Check for Advanced license
* **Checking for Extension Licenses:**
* if arcpy.CheckExtension("spatial") == "Available":
* arcpy.CheckOutExtension("spatial")
* You must **check licenses when running stand-alone scripts** outside of ArcGIS Pro.

**5.14: Accessing ArcGIS Pro Help**

* ArcGIS Pro **Help Documentation** is your friend!
* Find:
  + Tool descriptions and parameters
  + Example Python code
  + Visual illustrations of tool functions
* Access at:  
  <https://pro.arcgis.com/en/pro-app/help>

**Key Takeaways from Chapter 5**

✅ ArcPy is the essential bridge between Python and ArcGIS Pro.  
✅ Always check your license before running tools.  
✅ Use environment settings to maintain consistency in your outputs.  
✅ Pay attention to messages and warnings—errors often provide useful clues.  
✅ The **ArcGIS Help Documentation** is your go-to reference when writing scripts.

Yes, there are a few valuable insights from the **Week 10 (March 25, 2025)** class that would be **great additions** to your saved notes, especially since they reinforce your understanding of **debugging**, **error handling**, and practical project submission strategies. Here's what you should consider adding:

**✅ New Points to Add to Your Notes**

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

**🔧 Types of Errors**

* **Syntax Errors**: Mistyped code or formatting; script won’t run at all.
* **Exceptions**: Runtime failures; script starts but fails mid-process.
* **Logic Errors**: Script runs but gives the wrong result.

**🐞 Debugging Strategies**

* **Print Statements**: Insert after key blocks to confirm they ran.
* **Commenting Out Code**: Temporarily disable chunks to isolate where problems start.
* **Debugger Tools**:
  + Use breakpoints in PyCharm to step through code.
  + Check variable values line-by-line.
  + Use **Ctrl+C** to stop scripts if frozen.

💬 Mr. Bushland uses print statements himself and encourages commenting out **entire sections** to isolate bugs.

**🔍 Exception Handling Tips**

* Use try/except blocks to handle runtime errors gracefully.
* You can also use raise Exception("message") to create your own error triggers.
* Validate input and anticipate errors from user typos.

**📌 Common Mistakes to Watch For**

* **Incorrect slashes in file paths** (single backslash vs. forward slash vs. double backslash).
* **Mixed slashes** in one path (common student error).
* **Missing colons**, indentation errors, or wrong use of = vs ==.
* **Data not shared correctly** – especially when submitting .aprx files or shared project links.

**📁 ArcGIS Pro Submission Reminders**

* Share projects using **Share → Share Project → Upload to Online → Share with Organization**.
* Submit the **short URL** from ArcGIS Online → Content → Item → Details → Bottom right.

🔒 If you submit a local .aprx file instead, geodatabase paths often break due to differences in file structure or permissions.

**🧠 Real-World Cautionary Story**

* Mr. Bushland once ran a Python script as written by his boss.  
  It **renamed every file in every folder**, breaking links across the entire directory.
* Moral: Always understand **what a script is doing** before running it — especially on shared or production data!