Spring 2-1-2023

GPHY 491.01: Programming for GIS

K. Arthur Endsley
University of Montana, Missoula, arthur.endsley@umontana.edu

Programming for GIS

Geography 491/489 — Spring 2023

Tues & Thus, 09h30 - 10h50; Weds 09h00 - 10h50

218 Stone Hall (Tues, Thurs) & 106 Stone Hall (Weds)

Instructor:	**K. Arthur Endsley**, PhD, Research Scientist, NTSG
Email:	`arthur.endsley@ntsg.umt.edu`
Moodle site:	https://moodle.umt.edu/course/view.php?id=60666
Office hours:	Weds, 11h00 - 12h00, ISB 415

The contents of this syllabus are subject to change.

Course Overview

Resource managers, environmental scientists, and policy experts regularly ask questions that require spatial data to answer. They are fortunate that there is today a wealth of spatially explicit data on human communities and natural resources; but, sometimes, these datasets are so voluminous they can be difficult or impossible to work with in a desktop GIS or spreadsheet program. The questions we ask may also implicate spatial relationships (e.g., distance, proximity, routing) that can't be analyzed using built-in software routines. Moreover, when we conduct a spatially explicit study as a series of geoprocessing steps, how do we preserve and describe this recipe so that others can understand, verify, and *repeat* our analysis?

Learning Objectives:

- **Learn Python programming!** Python is consistently ranked within the top 3 general programming languages (2018-present, TIOBE index), and for good reason! It is easy-to-learn, scales up quickly, and supported by a large user community that contributes packages for a wide variety of applications. You'll also learn about R, which is a "meh" programming language but a powerful *analysis and visualization* tool.
- **Automate geospatial workflows.** When humans try to complete repetitive tasks (e.g., converting latitude-longitude coordinates into UTM coordinates), they frequently make mistakes. **Computers never make mistakes;** they carry out *our* instructions in a predictable and deterministic way. If you have a lot of data to process, clicking here, then there, then there again, in a desktop GIS, over and over, simply won't do!
- **Process large geospatial datasets.** Multi-decadal field campaigns, social media analytics, and satellite remote-sensing are increasing the volume and the frequency of data we collect. Desktop GIS and spreadsheet programs have limits—they simply cannot work with gigabytes or terabytes of data. You will learn to process "Big Data" by breaking the problem down into either separate tasks or subdatasets that can be processed independently and concurrently.
- **Document, explain, and reproduce your geospatial workflows.** You will learn to convert your GIS workflows into transferable and re-useable **scripts**. You'll learn about

literate programming, and how tools like Jupyter Notebooks and RMarkdown can help you combine code with narrative so that other people can understand, verify, and reproduce your work.

Prerequisite(s): You should be very familiar with a desktop GIS program (e.g., QGIS, ArcGIS) and understand what spatial data formats (e.g., raster, vector) and common spatial data file types (e.g., GeoTIFF, Shapefile) exit. You should also have basic familiarity with geographic and projected coordinate systems.

Textbook: There is no required textbook, but I can recommend some supplemental resources:

Think Python, 1st Edition, Allen P. Downey

Free to read at: https://greenteapress.com/wp/think-python/

Guide to NumPy, Travis E. Oliphant

Free to read at: http://web.mit.edu/dvp/Public/numpybook.pdf

Whirlwind Tour of Python, Jake Vanderplas

Free to read at: https://github.com/jakevdp/WhirlwindTourOfPython

Schedule at a Glance

Weeks 1 - 3: Introduction to Python Programming, *to prepare you for general-purpose programming in Python, including: file handling; input and output on the command line; writing and executing scripts; defining and applying re-usable functions; repeating tasks within loops; changing program behavior depending on conditions.*

Weeks 4 - 5: Raster Data Processing in Python, *where we will learn how to: read raster data from files in the Python environment; treat raster data as multi-dimensional arrays; write arbitrary arrays to raster data formats while handling spatial reference systems; combine multiple raster datasets; use multiple CPUs to process raster data concurrently; apply windowing functions/ generic filters to raster arrays.* **Applications:** Habitat suitability analysis, Land cover/ Land use classification, Landscape modeling (e.g., NDVI, TWI).

Week 6: Batch Processing and Scientific Computing in Python, *where we will learn to scale-up Python programs to process hundreds or thousands of spatial datasets in sequence, efficiently and accurately. We'll learn to write re-usable Python scripts that accept arguments or configuration files to define program behavior. We'll also learn about scientific data formats including HDF5 and NetCDF4.*

Week 7: Spatial Data Management with GDAL/OGR, *which will empower you to use efficient command-line tools for managing and manipulating both vector and raster data.* **Applications:** Bulk reprojection, Resampling, Mosaics

Weeks 8 - 9: Spatial Analysis in Python, *where we will briefly explore how to read and write vector data in Python, with the ultimate goal of using vector data (e.g., Shapefiles) to summarize raster arrays.* **Applications:** Zonal statistics, Census district enumeration, Proximity analysis, Landscape modeling.

Week 10 (Spring Break)

Week 11: Introduction to R for Data Analysis, *where we will treat R not as a programming language but as an environment for data analysis, focusing on tabular data (e.g., the attribute table of a Shapefile). This will lay the foundation for manipulating spatial data in R.*

Weeks 12 - 13: Vector Data and Spatial Analysis in R, *where we will learn to read and write vector data files in R; transform/ project data between different spatial reference systems; compute spatial metrics and combine these with attribute data.* **Applications:** Proximity analysis, Spatial autocorrelation, Spatial autoregressive models.

Week 14: Mapping in R, *where we apply our knowledge of managing and plotting generic data in R to spatial data, specifically, and produce regional, continental, and global maps that combine raster and vector data for effective communication.*

Weeks 15 - 16: Student Project Presentations

Course Policies and Expectations

- Arrive to class on time. Give me a heads-up if you have a consistently tough commute because of, for example, a class ending immediately before ours.
- Let me know if you're going to be late or absent. Things come up, and we're all busy. Help me to help you stay on track.
- During lecture (Tuesday-Thursday), keep your cell phones in your backpacks. Limited use on Wednesdays is fine, but devices should be in a quiet mode at all times while in the classroom.

Learning Procedures

During the Tuesday-Thursday classes, we will primarily use two types of learning:

- Hands-on, individual programming, where you reproduce the results of commands or small programs that I show to you or share with you.
- **Peer programming:** You and a partner will try to solve a programming challenge together: explaining to one another how a program works, debugging a program, or writing a small program.

How You Will Be Evaluated

- **Lab Exercises (70%)**
  - During Lab we will ask and answer **meaningful questions** about **real datasets**.
  - **Lab assignments are due 1 week after they are introduced;** i.e., they must be submitted before 10:00 AM on the following Wednesday.
- **Class Participation (10%)**
  - To make this easy and fair for everyone, I'll be tracking attendance. There are multiple valid reasons why you might be late or absent, so just let me know.
  - May also include completion of weekly surveys and Comprehension Checks, but is not based on correctness ("getting the right answer").
- **Graduate Increment: Midterm Project (10%)***
  - Students can choose to interview a GIS professional or conduct a software/ data review and present their findings to the class. *Detailed rubric will be provided after Week 1.*
- **Graduate Increment: Final Project (10%)***
  - Students will choose publicly available spatial dataset(s), present a new analysis of the data, and document and share their geoprocessing workflow. *Detailed rubric will be provided after Week 1.*

***Optional as extra credit for undergraduate students;** grade will otherwise be normalized by the Lab Exercises + Class Participation total.

Submitting Assignments

Assignments should be submitted through Moodle only.

- Unless otherwise announced by the instructor, **Labs are due by 9:00 AM the following Wednesday;** i.e., right before the next week's lab begins.
- Labs must be uploaded to Moodle as a compiled Notebook (`*.html` or `*.pdf` file) or as `*.ipynb` files.
- **The uploaded file should have your last name as the first part of the filename;** e.g., `Endsley_Lab1.html`
- Please try and simplify your Notebook before submitting it—just show the problem prompts, the code you wrote to solve it, and any relevant outputs. We're not using "templates" anymore because this was too confusing.
- If a question is asked as part of a Lab (beyond entering Python code), please use a Markdown cell to answer the question.

On Academic Dishonesty

You should be familiar with **the University of Montana Student Code of Conduct** and follow it accordingly. This includes Article IV, which prohibits plagiarism and other acts of academic dishonesty. I'm sure this won't be an issue, because we're all adults and we're all here to learn together!

Accessibility

We all have different ways of learning; some of us face challenges learning in or accessing a typical classroom environment. I'm here to help you succeed! For reasonable accommodation, please meet with me as soon as possible. The Office for Disability Equity (ODE) can assist both of us. For more information, visit the ODE website.

Required Software

I strive to make the course materials compatible with Windows 10+, Mac OS X, and GNU/Linux operating systems. This is more challenging than you may realize. The campus lab computers run Windows 10 with the "Windows Subsystem for Linux" (WSL), which enables them to run Linux programs, including the Unix Command Line. If you want to reproduce your work on your personal computer (and I highly recommend that you do), you'll want to set it up the same way or make sure it supports the same software.

For All Users

- **Install R and RStudio.** There are two steps: Install R, then RStudio. You'll find these steps, and the downloads, at this link.
- ***Follow additional instructions, below, for your system.***

For Windows Users

1. **Install OSGeo4W.** Go to: https://trac.osgeo.org/osgeo4w/ and "Download the OSGeo4W network installer." Run the installer, choosing "Express" install and choose to install GDAL and QGIS (if QGIS is not already installed).

2. **Start the OSGeo4W Shell** by typing "OSGeo4W Shell" in the Start or search menu.

3. **Install Python dependencies:** Type the following and hit ENTER:
   `pip install numpy scipy matplotlib pandas rasterio pyproj h5py netCDF4 pytest pysptools scikit-image scikit-learn line_profiler shapely geopandas`

4. **Install Jupyter Notebook:** Type the following and hit ENTER:
   `pip install notebook`

5. **Test that you can launch Jupyter Notebook:** Type the following, hitting ENTER at the end of each line; `username` should be replaced with your Windows username:
   ```
   cd C:/Users/username
   python -m notebook
   ```

For GNU/Linux Users

Instructions here are for Ubuntu, but Steps 2-4 should be the same on all GNU/Linux systems.

1. **Install the GDAL/OGR tools.** Launch a **Terminal** window and type:
   `sudo apt install python3-dev gdal-bin libgdal-dev python3-gdal`

2. **Install Python dependencies:** Type the following and hit ENTER:
   `sudo -H pip3 install numpy scipy matplotlib pandas rasterio pyproj h5py netCDF4 pytest pysptools scikit-image scikit-learn line_profiler shapely geopandas`

3. **Install Jupyter Notebook:** Type the following and hit ENTER:
   `pip install notebook`

4. **Test that you can launch Jupyter Notebook:** Type the following, hitting ENTER at the end of each line:
   ```
   cd
   jupyter notebook
   ```

Optional: Install a Python IDE

- **Install a text editor or Python interactive development environment (IDE) of your choice.** It's possible to complete all work in Python using the command line and a plain text editor (e.g., SublimeText, Notepad++). You might choose to install a dedicated Python IDE like Spyder, PyCharm, Geany, or LiClipse. Spyder can be installed in the Anaconda Navigator.

Tentative Schedule

Some of the recommended **resources** are ***required*** for participating in classroom discussions.

Class	Topics	Resources **(Bulleted items required)**
Tuesday, **Jan. 17**	**So, You're Programming Now?** • Motivating examples • Human-Computer interaction • Thinking like a computer scientist • Who is a computer scientist?	• "Stereotype threat: A summary of the problem" Planet Money: "When Women Stopped Coding" Downey (Ch. 1) Mattheis et al. (2022)
Weds., **Jan. 18**	**LAB** (Week 1) *Optional lab time for anyone who wants help installing or getting familiar with software.*	
Thursday, **Jan. 19**	**Introduction to Python Programming** • The Python interpreter • Variables and program state • Data types and Sequences • Functions and their application • Where to go for help	Downey (Ch. 2, Ch. 3 through Section 3.4) Intro to Jupyter Notebooks
Tuesday, **Jan. 24**	**Introduction to Python Programming (Cont.)** *Picking up where we left off on Jan. 19...*	Downey (Ch. 3, 5, 6 and Appendix A) Vanderplas, Ch. 3: Variables and Objects and Ch. 6: Built-in Data Structures
Weds., **Jan. 25**	**LAB #1** (Week 2) *Working with global primary productivity datasets to practice basic Python skills.*	

Thursday, **Jan. 26**	**Introduction to Python Programming, Part 2** • Comparison operators, the `is` and `in` keywords • Function arguments • Everything is an Object • Data structures • Loading Python packages	Downey (Ch. 3, 5, 6 and Appendix A) Vanderplas, Ch. 3: Variables and Objects and Ch. 6: Built-in Data Structures
Tuesday, **Jan. 31**	**Building Python Proficiency** • Python idioms: Sequences, Comprehensions • Control of flow • Debugging and Tracebacks	Downey (Ch. 7) Vanderplas, Ch. 9: Errors and Exceptions
Weds., **Feb. 1**	**LAB #2** (Week 3) *Building and applying Python functions for working with spatial features from airborne CO_2 flask data.*	
Thursday, **Feb. 2**	**Arrays and Plotting in Python** • Introduction to `numpy` • Multi-dimensional arrays in Python • Plotting data with `matplotlib`	Oliphant (Ch. 2) NumPy: The absolute basics for beginners NumPy Illustrated: The Visual Guide to NumPy
Tuesday, **Feb. 7**	**Raster Data Analysis in Python** • Reading and writing raster data in Python • The Geospatial Data Abstraction Library (GDAL) in Python • Affine transformations • Raster band math; Spectral indices • Multispectral rasters in Python	• The GIS&T Body of Knowledge: Python for GIS Unfamiliar with rasters? Read "Intro to Raster Data"

Weds., **Feb. 8**	**LAB #3** (Week 4) *You'll take gridded surface temperature data from NOAA, stored as an array, and export it to GIS-friendly formats. You'll also get some practice with querying raster array data and plotting histogram summaries of multispectral raster bands.*
Thursday, **Feb. 9**	**Raster Data Analysis in Python, Part 2** • Function application with raster data • Handling NoData • Array masks • Zonal statistics • Raster (re-)projection • GISGeography: What is Map Algebra?
Tuesday, **Feb. 14**	**Raster Data Workflows** • Introduction to `rasterio` • Reprojection with `rasterio` • Resampling with `rasterio` • Focal operations and `scipy` filters
Weds., **Feb. 15**	**LAB #4** (Week 5) *Using Landsat 5 data to monitor change in lake surface area.*
Thursday, **Feb. 16**	**Scientific Computing with Python** • Machine representation of numbers • Scientific provenance • Hierarchical Data File (HDF) format • NetCDF4 format • QA/QC Bit packing
Tuesday, **Feb. 21**	**Scaling-Up Raster Data Analysis** • Why is my Python code slow? • Concurrency vs. Parallelism • Parallel processing of raster data arrays
Weds., **Feb. 22**	**LAB #5** (Week 6) *Using MODIS surface reflectance data to monitor high-latitude vegetation trends.*

Thursday, **Feb. 23**	**Automating Tasks with Python** • Writing Python scripts • Python at the Command Line • The Python Module Pattern • Reading files	Downey (Ch. 10-12, 14, 15)
Tuesday, **Feb. 28**	**Scientific Computing with Python, Part 2** • Creating command-line interfaces • Separating code from data with configuration files • Verification and Validation (assertions and tests) • Reproducible workflows	Downey (Ch. 4) • Wilson et al. (2014): Best practices for scientific computing
Weds., **March 1**	**Optional LAB #6** (Week 7): **GDAL-OGR Workflows** • Introduction to the Command Line • Navigating files and folders • Programs and arguments • `gdalinfo` and `ogrinfo` • `gdalwarp` and `gdal_translate` • Clipping rasters with `gdalwarp` and cutlines • Creating mosaics with `gdal_merge.py` • Raster math with `gdal_calc.py` • Vector data management with `ogr2ogr`	
Thursday, **March 2**	**Spectral Analysis** • Mixing spaces • Principal components analysis (PCA) • Dimensionality reduction • Spectral Angle Mapper	

Tuesday, **March 7**	**Vector Data Analysis in Python** • GeoJSON and `json` • Opening geometry collections with `fiona` • Geometric operations in `shapely` • Coordinate transformations • Introduction to `geopandas`	Unfamiliar with vector data? Read "Intro to Vector Data"
Weds., **March 8**	**LAB #7** (Week 8) *Using a Shapefile of burned-area boundaries, practice vector data processing and computing geometric relationships using* `fiona`, `shapely`, *and* `geopandas`	
Thursday, **March 9**	**Vector Data Analysis in Python, Part 2** • Spatial joins in `geopandas` • Extracting raster values using Point or Polygon geometry • Zonal statistics	
Tuesday, **March 14**	**Introduction to Statistical Learning** • Classification versus Regression • Training and testing data • `scikit-learn` • Naive Bayes classifier	
Weds., **March 15**	**LAB #8** (Week 9)	
Thursday, **March 16**	**Midterm Project Presentations**	
Tuesday, **March 28**	**Introduction to R for Data Analysis** • Reading tabular data in R • Selecting columns, filtering rows • Pipelines with `dplyr` • Introduction to RMarkdown	

Weds., **March 29**	**LAB #9** (Week 10)	
Thursday, **March 30**	**R for Data Analysis, Part 2** • Pivoting and reshaping with `tidyr` • Beginning plotting with `ggplot2` • Effective visualizations • Perceptually linear color scales	Rougier et al. (2014): Ten simple rules for better figures Ware (1988): Color sequences for univariate maps
Tuesday, **April 4**	**Vector Data in R** • The Simple Features library • Plotting simple features with `ggplot2` • Feature collections and their attributes • Projections and coordinate reference systems • Text data to spatial data with `st_as_sf()`	
Weds., **April 5**	**LAB #10** (Week 11)	
Thursday, **April 6**	**Raster Data in R** • The `raster` library • Rasters as Data Frames: Why, God? • Plotting raster data with `ggplot2` • Multi-band raster data in R • Raster map extent • Projecting raster data	