Advanced - Python

EECS 201 Spring 2026

Submission Instructions

This assignment is an "online assignment" on Gradescope, where you will attach your files and answer some questions.

Presentation/Checkoff

In addition to submitting this assignment, you will need to attend a lab session after the deadline to discuss your work with the course staff. Refer to the course schedule for the window for doing so for this assignment.

As part of the discussion, be prepared to do the following:

- Explain your approach for the assignment. What documentation did you read, and how did you find it? What did you try that didn't work, and what was your process for figuring out how to fix it?
- Summarize what you learned from the assignment.
- Discuss how you plan on implementing the tools in this assignment and what you learned in your future projects.
- Apply your approach for solving the problems in this assignment to a new problem that you have not seen before.

I suggest keeping these in mind while doing the assignment, and writing down your thoughts so that you can review them before the checkoff.

Your overall score for the assignment will be a combination of your submission score and the presentation/checkoff score. Refer to the syllabus for the details.

1 Who needs MATLAB? (5)

It was mentioned in lecture that you could probably replace MATLAB with Python + NumPy + SciPy + Matplotlib for general purpose use (unfortunately, MATLAB does have some really nifty tools in more focused areas that can be hard to replace).

For this exercise, we're going to keep it simple since the SciPy stuff tends to be more domain specific: no need to work on FFTs, as not everyone has taken EECS 316. The only package dependencies you'll need are NumPy (for data types) and Matplotlib (for plotting). If you don't have those, try installing the `pip` Python package manager and installing those two packages (or do whatever setup you need to install Python packages for you particular system). You might also want to look into "virtual environments" for Python.

Write a **Python script** that plots $y = x^3 - 7x^2 + 2x + 2$ for $-4 \leq x \leq 4$ with the title "Who needs MATLAB?" and saves the figure to a PNG file called `output.png`. The number of samples you use is up to you, but the figure should not look jagged from a low sample count. This must be a **Python script**: no Jupyter notebooks and the like will be accepted.

- ☐ Submit your Python script file.
- ☐ Submit the plot you generate.
- ☐ Mention how much previous experience you have had with MATLAB, NumPy, and Matplotlib.

Helpful hints:

- For those unfamiliar with the general MATLAB workflow for generating plots, you'll need to generate samples for your X coordinates and samples for your Y coordinates: these will be matched up by index.
  For example, your X coordinates could be [-4, -2, 0, 2, 4] and your Y coordinates could be [16, 4, 0, 4, 16] when plotting $y = x^2$, albeit with only 5 samples.
  These samples would then be passed to a plotting function.

- NumPy comes with its own version of MATLAB's `linspace()`: `numpy.linspace()`. This can come in handy for generating your X coordinate samples.
- If you do this right, given that you have your X coordinate samples, you only need one line to calculate your Y coordinate samples. `numpy.ndarray`s are your friends!
- Matplotlib's MATLAB-like plotting function is `pyplot.plot`. `pyplot` also has a mechanism to save figures.

2 Go wild! (5)

The Python standard library is pretty darn big. There are many sorts of problems that you face in each of your classes or in your life, and there's a good chance that some of them you can solve with a Python script(s) and using the Python standard library. Or perhaps the standard library is missing some feature (eg. YAML parsing) and you need to install some package.

The point of this particular exercise is to go wild and stretch out your wings with real problems you face: this is probably the best way to learn and get a feel for Python.

Minimum requirements are to use at least one of the more "advanced" modules/packages and have either some sort of file I/O or networking capability. Modules that provide fairly basic/fundamental things like `os`, `math`, `string` don't count, nor do the modules used in the first part of this assignment; feel free to ask if your choice in "advanced" module is appropriate. In addition, your script should do something reasonably useful: being a simple example of calling some functions from a module will not suffice (e.g. simply reading and printing things out from a CSV file, or computing basic statistics like mean and standard deviation: your script should do *something meaningful* with the data). *Some* examples of modules/packages that I find "advanced" enough from the standard library: `csv`, `argparse`, `curses`, `email`, `json`, `html`, `xml` (and friends), `http.client`, `sqlite3`, `tkinter`.

Do **not** submit something you did for another class or wrote previously. The purpose of this assignment is to learn something new, so we will not give credit for things that don't meet that goal.

- ☐ Submit your Python file(s) and any additional data files needed.
- ☐ What packages did your script depend on?
- ☐ What is your script for and what does it do?

Ways to get points

These build off each other: later items require earlier items to be fulfilled.

- Shows the use of a function(s) of some "advanced" module(s): 2 points
- The above, and performs a "useful" task with the module(s): 4 points
- The above, and has some run time configurability (e.g. arguments, config files, etc.) so behavior can be changed without needing to edit the script: 5 points
- The above, and the scope exceeds expectations for an advanced assignment and/or is particularly creative: 6+ points

Examples of what is expected:

- A configurable mass emailer
- A data analysis tool that gives a meaningful result (not just basic statistics)
- Something that can automate some task for you
- A small game

Not quite there:

- Printing out "Hello world" with some UI library
  To build off of this, add some logic so that the UI application can do something that can aid in life.
- Tossing random generic data into a Pandas DataFrame and doing a basic operation
  To build off of this, have the data be less generic and have some sort of meaning to it (e.g. food nutrition, spending, etc.) and then do *something meaningful*, something useful to some person who'd run this script.
- Iterating through a CSV file and printing out each row
  To build off of this, do something with the data in the CSV file (e.g. maybe the CSV file can represent contact information to send emails with, and your program sends an email to each address)
- Making an arbitrary web request to some API
  To build off of this, have the web request be of some purpose (e.g. get a cat fact, retrieve some data to be processed) and meaningful complexity (e.g. use Beautiful Soup to parse a web result)

- A grade calculator
  To build off of this, add more complex functionality to your script (e.g. retreive grade data from the Canvas API)

If you have trouble coming up with something, think about your daily life. What small annoyances do you have that you could potentially solve with some sort of automation? Maybe you have numbers to be crunched, files to be renamed, files to be generated, or data to be transformed from one format to another?