Human-AI Team Programming for METOC (MR2020; Summer Quarter 2024)
Officially Computer Computations for Air-Ocean Sciences
Instructor: Scott Powell (Root 255)
Room: IDEA Lab (Root 123)
Meeting times: T, W: 1100–1230
Course webpage: https://swpowell.github.io/MR2020.html

Course Objectives

- Learn basics of programming data structures and control flows, object oriented programming, and data visualization
- Learn to develop code while teaming with generative AI
- Familiarization with state of the art software development platforms and environments

Syllabus

The syllabus is approximate. We will plan to cover all material listed, but the timing may differ from the schedule depending on the learning pace of the class. In addition, there may be material covered that is not listed on the syllabus depending on the needs of students and on questions raised during class.

Week 1 (July 9, 10): Introduction to UNIX-based OS. Getting setup on IDEA lab or personal computers with miniconda + python environments, VSC, GitHub, and GPT. Features and integration of VSC and GitHub. Comparison with MATLAB.

Homework: Review Visual Studio Code Introduction
Basics of VSC: (https://code.visualstudio.com/docs/introvideos/basics)
Interactive Python in VSC: (https://code.visualstudio.com/docs/python/jupyter-support-py)

Week 2 (July 16, 17): Querying GPT. Adding software to python environments. Running simple codes in VSC. Numpy arrays and functions. Data Types (numbers, strings, lists, tuples, sets, dictionaries). Indexing. Broadcasting. Module imports. Print functions. Manually debugging code with interactive python.

Week 3 (July 23, 24): Control structures: If-else blocks. For and while loops. Try-except blocks. Match-case blocks. List comprehension.

Homework: Problem Set 1 (Due July 31)

Week 4 (July 30, 31): Functions. Classes. Placing functions and classes in separate code and calling that code as a module. Understanding input and output parameters. Errors and Exceptions.

Week 5 (Aug. 6, 7): File input/output using xarray. Introduction to pandas. Parallel coding using multiprocessing module.

Homework: Problem Set 2 (Due Aug. 13)

Week 6 (Aug. 13, 14): Midterm week. Review. The exam is written. Students will not be able to use GPT on the exam.

Midterm Exam: Aug. 14 at 1100

After the midterm, students will begin working on their final projects, which should be selected by Aug. 20.

Weeks 7–11 (Aug. 20 – Sept. 17): Understanding logical flow. Continue learning to properly query GPT to generate code. Debug and comment code. Create figures and subplots, making line plots, scatter plots, histograms, two-dimensional plots. Plotting geospatially located data on maps. Labeling axes. Creating legends and color bars. Teaming with GPT to generate and revise plots.

Grading

Midterm (50%)
Final practicum (50%)

All students will have two opportunities to take the midterm exam. Any student who is not satisfied with their grade on the first try will be allowed to attempt a second different exam covering the same topics. The highest of the two grades will be the one recorded.

Approximate grading scale (after highest grade is curved to 100*):
These are the minimum letter grades students can receive if their numerical grade is within the range shown in the left column. For example, a student with a final grade of 90 will have earned no lower than a A-.

Numerical Grade	**Minimum Possible Letter Grade**
93–100	A
90–92	A-
87–89	B+
83–86	B
80–82	B-
77–79	C+
73–76	C
70–72	C-
67–69	D+

*The difference between 100 and the highest student grade at the end of the course will be added to each student's grade before being converted to a letter grade. For example, if the highest course grade is 93, 7 points will be added to each student's grade. The new total with the additional 7 points will then be converted to a letter grade.

63–66	D
60–62	D-
Below 60	X/F

Course Structure:

1. Course material will be available at or linked from the course webpage.

2. Students may also reach the instructor via private chat on Teams or email.

3. Students will pair in randomly assigned teams during the quarter and will work in these same teams for the final project. Students who wish to work individually should indicate so at the beginning of the quarter.

4. The midterm exam is intended to test student comprehension of basic syntax, data structures, and logical control flows. These are essential skills that are required to understand coding in any language and are needed to effectively team with GPT to produce code. Therefore, use of GPT is not permitted for the midterm. However, students are free to team with any AI tool of their choice to build the code for their final project.

5. Students will select (with instructor guidance, as needed) the problem for their final project. By the date indicated in the syllabus, students should develop a plan for the final project. All elements of the plan must be approved by the instructor before work begins. The elements include:

    a. Identification of the problem: What question(s) is/are to be answered? What is to be visualized?
    b. Required data: What data are needed to address the problem? Where are the data going to be acquired? What format are the data available in? How big are the data? Does the data provide an I/O constraint? If so, is it surmountable?
    c. Analysis: How will the data be processed or analyzed? What steps are needed to convert the data from the form in which it is read to the quantity or quantities you want to visualize? **NOTE**: This is the core of the project. A project is not to simply download data and make a plot. You need to do something with the data!
    d. Visualization: How will the results be visualized? What types of plots and color schema are most appropriate?

    Each of the elements may evolve as students embark on executing the plan; however, this rough outline should be decided upon before starting the final project.

6. The final project will be graded based on the following:

- Did the student generate a correct final product/output (including visualizations) and can the code be run by the instructor? (50%)
- Is the code readable, properly structured (i.e., proper use of functions, etc.) and well-commented? (30%)
- Is the code efficient (i.e., run time and memory usage minimized)? (20%)