CMSI 1998

SPECIAL STUDIES: SOFTWARE CARPENTRY*

http://dondi.lmu.build/spring2026/cmsi1998

Spring 2026—Pereira 121 MW 1:40–2:50pm (01) John David N. Dionisio, Ph.D.
Seaver 304 MW 4:30–5:40pm (02); 2 semester hours *dondi@lmu.edu*

Office Hours TR 2–5pm, Doolan 102
or by appointment *(don't hesitate to ask!)* (310) 338-5782

Objectives and Outcomes

*The phrase "software carpentry" was coined by Greg Wilson and Brent Gorda in 1998:

https://software-carpentry.org/

This is a two-unit practicum course, meaning you will receive training and gain experience building systems according to modern best practices. This particular practicum focuses on the *tools* that you use for programming and not the programming itself. Furthermore, we will predominantly use the *command interaction style*—widely considered to be the most efficient way to get work done…but the hardest one to learn. It will be worth it though! Long after the course concludes, my hope is that you will be able to:

1. **Show proficiency with the command line interaction style**
2. **Manage, navigate, and modify files exclusively within an operating system shell**
3. **Perform version control operations exclusively with commands**
4. **Write and manage code collaboratively, including but not limited to resolving conflicts due to concurrent work**

In addition to the course-specific content, you are also expected to:

5. **Follow disciplinary best practices throughout the course.**

Prerequisites/Prior Background

You should have done some computer programming before—and in fact, you'll be asked to revisit some previously-written programs, so have those ready. Most importantly, you must bring a growth mindset and an openness to new and unfamiliar ideas. This course involves activities that are unique to software development, and as such are outside the realm of typical technology interactions. Mindful practice and repetition will be key.

Materials and Texts

Most of the course content will be delivered live in class, immediately followed by hands-on practice and repetitions. The same information can be found online in many forms, but may be intended for a different audience and varies highly in quality and timeliness. As such, the recommendation is to encounter the content in class first, then use online resources as references and/or supplements.

As a practicum, we focus on specific technologies to keep things cohesive and concrete. For this offering, we will be using commands that follow the conventions of Unix and Unix-style operating systems. The concepts do transfer over to other technologies such as PowerShell or specialized systems like databases and analytical tools, and I will aim to point out these underpinnings as we go. For in class activities and assignments, you'll need:

- A terminal/command-prompt program which supports Unix-style commands. Linux and macOS come with this already; on Microsoft Windows, Git Bash and Windows Subsystem for Linux (WSL) are viable (and free) options
- A "supercommand" that we will explore in depth is *git*—the current de facto version control software in industry today. On Linux, installation depends on the distribution you have; on macOS, *git* comes with the *Command Line Tools for Xcode* package; on Windows, Git Bash includes the *git* command already
- The nature of the course work (see below) will involve recording yourself and your screen on video. For this, Zoom will suffice, but other apps will also do as long as they create video files in a standard format and encoder

Course Work and Grading

This course focuses on skills, so deliverables predominantly involve *recording* yourself in action. For additional cohesiveness, the assignments are

framed as episodes of a podcast or video channel—who knows, you might be a highly effective tech communicator just waiting to be discovered! As an incentive, additional credit may be given for videos with an appreciable "fun factor" or creative premise. Do take note that although entertainment value makes informative videos better, it does not redeem uninformative ones. This will be reflected in the way assignments are graded.

Your final grade will be based on the percentage of the points you get for the following deliverables against the total number of possible points:

Filescape navigator	100
Curator personalis	100
Edits, envs, and execution	100
To [Git]hub and to hold	100
The non-accidental programmer	100
Dynamic duo	200
Welcome to the multiverse	200
Use the force you should not	100
Total	1000 points

Percentages ≥ 90% get an A– or better; ≥ 80% get a B– or better; ≥ 70% get a C– or better. I may nudge grades upward based on qualitative considerations such as degree of difficulty, effort, class participation, time constraints, and overall attitude toward the course.

Disciplinary Best Practices (Objective 5)

Appearance of these in the syllabus constitutes fair warning of the consequences of not heeding them.

Punctuality

An assignment's (episode's) number is its due date in *mmdd* format, and it is always due by 11:59:59.999pm of that date. If you primarily work in the Keck Computer Science lab, make sure to plan on finishing at home or a few hours earlier than that, because the lab isn't officially open that late so access is not guaranteed.

Point values are based on the state of your assignments at that moment. If there is an issue that prevents you from submitting an assignment on time (e.g., personal or family issues, sickness, conference attendance, can't-miss raid battles, job interviews, a family trip, or emergencies), let the instructor know ahead of time.

Version Control

Version control is an indispensable part of today's computer science landscape in industry, the academe, and the open source community. Effective use of version control is a focus of this course, with the intent that *you will demonstrate these good habits in future semesters and beyond.*

Such habits include but are not limited to: steady, incremental progress rather than "one and done" commits; descriptive messages showing mindful units of work; and the ability to collaborate with others and resolve conflicts in overlapping work. Make sure to acquire these habits not solely to do well in this course, but also to learn a key best practice within our discipline.

Workload Expectations

In line with the *LMU Credit Hour Policy*, the workload expectation for this two-credit-hour course is a minimum of $2 \times 3 = 6$ hours of work by an average student per week, including the time that we spend together in the classroom.

Attendance

Attendance at all sessions is expected, but not absolutely required. If you must miss class (for reasons such as those listed in the *Punctuality* section, or anything else), it is your responsibility to notify me about this and keep up with the course.

The last day to add or drop a class without a grade of W is January 16. The withdrawal or credit/no-credit deadline is April 10.

Academic Honesty

Loyola Marymount University is a community dedicated to academic excellence, student-centered education, and the Jesuit and Marymount traditions. As such, the University expects all members of its community to act with honesty and integrity at all times, especially in their academic work. Academic honesty requires that all members of the LMU community act with integrity, respect their own intellectual and creative work as well as that of others, acknowledge sources consistently and completely, act honestly during exams and on assignments, and report results accurately. As an LMU Lion, by the Lion's Code, you are pledged to join the discourse of the academy with honesty of voice and integrity of scholarship.

Academic dishonesty will be treated as an extremely serious matter, with serious consequences that can range from receiving no credit for assignments/tests to expulsion. It is never permissible to turn in any work that has been copied from another student or copied from any source (including the Internet) without properly acknowledging/citing the source. It is never permissible to work on an assignment, exam, quiz, or any project with another person unless your instructor has indicated so in the written instructions/guidelines. It is your responsibility to make sure that your work meets the standard of academic honesty set forth in the "Academic Honesty Policy" found at:

https://academics.lmu.edu/honesty

Technology Use and Academic Honesty

In this course, academic honesty includes the *appropriate* use of technology as an aid for learning and productivity. This includes but is not limited to generative artificial intelligence tools such as ChatGPT and GitHub Copilot.

Modern Generative AI provides fast solutions to a variety of computing problems, but should be used responsibly in the classroom setting to get the most out of your education. Generative AI will not always be helpful nor available, like in job interviews, technologies it hasn't seen, or extraordinary/unique scenarios. It cannot independently validate the code that it produces, which may contain bugs or security vulnerabilities. As such, you will still need to deeply understand the foundations of computing and should use it in your coursework with care, especially when its use might rob you of the same learning and desired difficulty of problem solving that comes from programming on your own.

That said, here are some acceptable use cases of generative AI on programmatic assignments:

- Generating docstrings and type hints
- Getting examples/explanations for language mechanics/syntax (e.g., how to use the spread operator; how to assign variables directly from an array or object; how to implement an error boundary component)
- Interpreting error messages
- Generating unit tests for edge cases
- Brainstorming new ideas for apps or app names

TL;DR *Never use technology (including but not limited to AI) to avoid learning, but do use it to augment, amplify, or accelerate (your) learning.*

Special Accommodations

The Disability Support Services (DSS) Office offers resources to enable students with ADD/ADHD; physical, learning, and psychiatric disabilities; and those on the autism spectrum to achieve maximum independence while pursuing their educational goals. Staff specialists interact with all areas of the University to eliminate physical and attitudinal barriers. Students must provide documentation for their disability from an appropriate licensed professional. Services are offered to students who have established disabilities under state and federal laws. DSS personnel also advise students, faculty, and staff regarding disability issues. Students who need reasonable modifications, special assistance, academic accommodations or housing accommodations should direct their request to the DSS Office as soon as possible. All discussions will remain confidential. The DSS Office is located on the 2nd floor of Daum Hall and may be reached by email at *dsslmu@lmu.edu* or phone at (310) 338-4216. Please visit *http://www.lmu.edu/dss* for additional information.

Course Evaluations

Student feedback provides valuable information for continued improvement. All students are expected to fairly and thoughtfully complete a course evaluation for this course. This semester, course evaluations will be administered online through the Blue™ evaluation system. You will receive an email notification at your Lion email address when the evaluation form is available. You may also access the evaluation form on Brightspace (*https://brightspace.lmu.edu*) dashboard during the evaluation period. Your responses will be anonymous and will not be linked to you in any way.

Topics and Important Dates

Specifics may change as the course progresses. University dates (italicized) are less likely to change.

January	Command line fundamentals: files, folders/directories, autocomplete, history, common shortcuts
January 16	*Last day to add or drop a class without a grade of W*
January 19	*Martin Luther King, Jr., Day; no class*
February	Manipulating files on the command line; editing within a terminal application; environment variables; introduction to *git*; core *git* flows and subcommands (init, clone, add, remove, commit, push, pull)
March	Introduction to Markdown; monitoring how code changes (log, diff); advanced code management (restore, amend, stash, reset, checkout)
March 2–6	*Spring break; no class*
March 31	*Cesar Chavez Day; no class*
April 1–3	*Easter break; no class*
April	Collaboration and merge conflict resolution (switch, branch, merge); the "red zone:" rebasing, forced pushes, etc. (i.e., with *git* power comes *git* responsibility)
April 10	*Last day to withdraw from classes or apply for Credit/No Credit grading*
May 11	*Final podcast "screenings"*

You can view my class calendar and office hour schedule in any iCalendar-savvy client. Its subscription link can be found on the course website (it's too long to provide in writing).

Tentative Nature of the Syllabus

If necessary, this syllabus and its contents are subject to revision. Students are responsible for any changes or modifications announced or distributed in class, emailed to students' LMU Lion accounts, or posted on LMU's course management system, Brightspace. If you are absent from a synchronous class session, it is the student's responsibility to check Brightspace and with the professor to see if you missed any important class announcements. Students should not rely on word-of-mouth from classmates.