Software Design Lab

DHUM 71000
Course Number: 59977
Wednesday, 6:30 - 8:30 PM
3 Credits, Room 3309

Patrick Smyth
patricksmyth01+softwaredesign@gmail.com

Dropbox readings)

Course Description

Many digital humanities projects require the creation of software, and many of these projects are large, complex, or require sustained collaboration. Knowledge of particular methods, processes, and tools is necessary for completion and maintenance of significant projects in the digital humanities. This course will give students a foundation in software development methodologies that they can draw from throughout their coursework and career.

This is a technical course, and students will learn a variety of hard and soft skills important for successful project completion. These include a limited number of fundamental concepts in programming, the use of version control, common software design patterns, managing state and persistence, and the basics of test driven development (TDD). The course will focus on two software "stacks," or collections of systems and tools frequently used alongside one another: a WordPress stack less focused on writing code, and a flexible stack based on coding in the Python programming language. Broader topics of discussion will include working to specifications, time line estimation, formulating an MVP, using project management tools, reading documentation, building for maintainability, and software ethics. After completing this course, students will be able to evaluate tradeoffs in software design, collaborate in a small group of mixed skills, and implement the most common techniques for designing modern software.

Our Learning Objectives

In this course, we'll

- Learn what it means to design software.
- Practice fundamentals of programming in the Python programming language.
- Engage with the humanistic side of software development, including considerations of semantics, ethics, accessibility, and representation.
- Consider selected topics in the history of computing.

- Develop literacy in tools relevant to developing software, such as Git and the command line
- Explore domain-specific languages such as HTML, CSS, and SQL
- Identify programming paradigms such as imperative, object-oriented, and functional programming.
- Learn industry techniques for collaboration, specification, planning, and negotiating technical tradeoffs.

Accessibility and Accomodations

If a reasonable accommodation by myself or the Graduate Center would help you to achieve your goals for this course, you may either approach me for a discussion of possible accommodations or speak with the Graduate Center's Disability Services Office. You can connect with Disability Services staff by emailing disabilityservices@gc.cuny.edu. Universal access is a critical part of my pedagogy, and I am always willing to discuss accessibility improvements, whether or not the discussion is predicated on a formal recognition of a disability by the university.

Patrick's Technical Philosophy

It's All Just Text

Most of what we'll do will involve writing text and handing it off to the computer to do stuff with it. This has some implications that I look forward to sharing with you.

Boring Is Better

I tend to prefer technologies that have been around awhile. That means they're stable, there are copious and high-quality learning resources, they'll stick around, and they've been tested in the fires of production. (This is also why I like reading novels from before 1900.)

Simple, Not Easy

Simple means that something does one thing and is therefore comprehensible. Easy means someone else did it for you, and guessed how you'd want to use it. The `cat` program you will learn the first day is simple. Facebook is easy.

Build from Small Pieces

It often makes sense to understand small pieces of functionality really well, then make them talk to each other in ways that won't be confusing to you or to others.

Practice, but Practice

Many people, including myself, have found that one learns technical skills most readily by alternating between performing deliberate practice, such as following a tutorial or reading a book, and doing practical things, such as working on small projects. You might also call this practice and praxis.

Our Greatest Foe, Complexity

We shall fight it on the beaches. We shall fight it in the corridors on the fifth floor. We shall fight it in the kitchenette. We shall not flag or fail. We will fight complexity wherever we find it.

Teaching Is Learning

To the extent that I know anything on these subjects, I know them because I've taught them. I encourage you to share your knowledge early, often, and freely. Develop your skills with an eye toward giving back to your community.

Assessment

15% Discussion and Lab Contribution 15% Personal goals follow-through 20% Lab Journal 20% Client role OR white paper 30% Group software design project

Discussion and Lab Contribution (15%)

This portion of your grade is based on your participation in weekly discussions and technical labs.

Personal goals follow-through (15%)

At the beginning of the semester, we will set a series of specific personal goals based on expertise we wish to develop over the course of the semester, as well as a benchmark for how to demonstrate that expertise. The grade will therefore be based on your ability to meet your own expectations. This portion of the grade

is intended to motivate us to seek what we need from the required self-study necessary for the course.

Lab Journal (20%)

Your personal lab journal is an honest reflection of your own individual technical study. Every week, you will write the amount of time you spent on technical learning and experimentation and thoughts on your progress. The journal is intended as a mechanism to provide credit to you for time-consuming technical work, as well as to allow you to calibrate your habits and find out what approaches to study are most effective for you. To encourage the honesty needed for the journal to be useful, you may take two health weeks where you record no entries.

Client role OR white paper (20%)

At the midpoint of the course, all students will suggest a well-scoped (that is, small) project, not to complete themselves but to be completed by a team of three fellow students. I will choose a set of these projects, and those students will serve in a client role for a team of developers. As the client role will take time, these students will be exempt from the short white paper assignment.

The white paper will take the form of a long (2000 words) blog post, academic paper, or Jupyter notebook that will be a reflection or development of a topic presented in the course.

Group software design project (30%)

The group software design project takes the form of a collaboration to complete a software project specified by another student taking a client role. You will go through a process of communicating with your client, developing requirements, negotiating technical tradeoffs, planning an architecture, dividing up the technical work, meeting client requirements, and putting the result into production.

Schedule

Week One

Homework: Get Ready to Bash

In our first session, we will be discussing the aims and shape of the course. In the second half of the session, we will be learning some basics of the command line and Git. Your homework for this first day, should you happen to manage to read this syllabus in time, is to prepare your computer for this lesson by following the below instructions:

Installing Git and Git Bash

Discussion: Welcome, Course Overview, and Guiding Principles

For our discussion this week, we will review the course and this syllabus.

Lab: Command Line and Git

For our lab this week, we will explore two foundational tools for modern software design: the UNIX command line and the Git versioncontrol system.

Optional Lab Resources:

DHRI Command Line tutorial
Command Line Crash Course (from Learn Python the Hard Way)
Programming Historian Command Line tutorial
DHRI Git tutorial
Version Control Before Git with CVS

Week Two

Homework

Journal: Course goalss
Read *Decoding Liberation* Chapter 1: A Brief History of Computing and Software Development
Read *The Design of Design*Chapters 1, 2 and 3
Read the Agile manifesto

Discussion: Design Philosophies

In software design, the most difficult problem is not in coding, but in planning, organizing, and completing software projects that match to concerns in the real-world. This week, we will compare prevalent design philosophies, including "waterfall" and "agile," and ask what we can learn from industry software development to apply in a digital humanities context.

Lab: Plain text

In this lab, we'll talk about plain text—what it is and what it can be used for. We'll cover the use of a text editor, writing markdown, working with plain text in the command line, and pushing files to GitHub.

Optional Lab Resources:

Plain Text Definition
Original post introducing markdown
What Is Markdown, and Why Is It Better for My To-Do Lists and Notes?
Sustainable Authorship in Plain Text using Pandoc and Markdown

---

Week Three

Homework

Install Anaconda
This will be a full lab session, with no discussion this week. Please prepare yourself for learning Python by spending time with the optional lab resources.

Lab: Python I

Our first session on the Python programming language will include these topics:

Using the REPL
Data types
Variables
Running Python scripts
Boolean logic
Conditionals
Iteration: for and while loops

Optional Lab Resources

Learn Python the Hard Way — Exercises 0 to 6
Python Programming/Creating Python Programs (A quicker but less detailed guide to running Python programs, may be useful if you've programmed in other languages)
What Is Python Used For?

Week Four

Homework

Read *A Web for Everyone*: Chapters 1 and 2 How People with Disabilities Use the Web: Tools and Techniques (W3C)
Dear Developer, The Web Isn't About You
Two Bit History: The World Wide Web and Its Inventor

Discussion: Accessibility

Accessibility is the quality of objects, interfaces, and information that determines their availability to a broad variety of human experiences. If software is inaccessible, it may only be usable by a narrow band of humanity. Since an overwhelming majority of software developers are young and able (also white, economically advantaged, and male), considerations such as users with alternate cognitive, sensory, and mobility needs or infrastructural inequality are considered an afterthought. This week, we will discuss access in software design and specifically the web, including consideration of semantics, fallbacks, and "simple vs. easy" in software design.

Lab: HTML/CSS

HTML (HyperText Markup Language) is a way to meaningfully describe documents and their various elements in plain text for representation on the web. CSS, or Cascading Style Sheets, is a language for describing how HTML documents should be presented to users. In this session, we will

Optional Lab Resources:

HTML on W3Schools
CSS on W3Schools
HTML and CSS on Khan Academy
Javascript for Cats

Week Five

Homework

Why Use a Static Site Generator?
Creating a website with WordPress: The good and the bad
What is Bootstrap? – The History and the Hype
Brutalism: The 'ugly' web design trend taking over the internet

Install Jekyll

Discussion: A Practical History of the Web

In the beginning, there was HTML. Web pages were documents that connected to other documents, and were created by hand-writing HTML. Later, in the Golden Age of Blogging, content management systems like WordPress rose to

popularity by providing database-powered software to dynamically generate HTML whenever a visitor requested a page. Today, tools like Jekyll provide a third path that provide advantages over CMS in specific situations.

Also, Javascript. And web apps. And frontend frameworks. How does all this stuff fit together?

This discussion will describe web tecnology in 2018 by tracing some history and taking note of major concepts and movements.

Lab: Static sites with Jekyll

Static site generators provide a way to build a website that has modern functionality—blogging, RSS, templating, markdown—without the overhead of a database. While certain features, such as comment sections, are difficult to implement on a static site, static sites have many advantages: they are more maintainable, simpler to understand and modify, are difficult to hack, and are more archivable and sustainable. They also fit neatly into a content management workflow that includes the command line, a text editor, Git, and markdown.

Week Six

Homework

Decoding Liberation: Chapter 2
FSF: What Is Free Software?

And choose and read one of the following:

The Cathedral and the Bazaar or Free as in Freedom

Discussion: Ethics of Free, Open Source, and Proprietary Software

Lab: Writing Functions in Python

Update: Here is the application we wrote in class: Magic 9 Ball

---

Week Seven

Homework

Read An Introduction to Programming Paradigms

Discussion

In this session, we'll use our discussion time to introduce the idea of a programming paradigm and two paradigms in particular: functional programming and object-oriented programming.

Lab

In this lab, we'll introduce some new concepts. First, we'll talk about complexity and why we avoid it in programming. Then we'll discuss "state"—keeping track of changes over time—and how different styles of programming deal with it. We'll try to solve a problem usingb three different paradigms: imperative, functional, and object-oriented.

Optional Lab Resources:

Functional Concepts
Object-Oriented Concepts

Week Eight

Homework

Continue work on journals. No formal reading.

Discussion: Iterating on a Prototype

Visitors: Designers Adam Brodowski and Clara Bunker

In designing software, one must typically work with clients or constituents to delineate and refine requirements. In this discussion, Clara Bunker and Adam Brodowski, two design professionals who specialize in product iteration and rapid prototyping, will join us.to describe the process of working with clients and iterating on a design.

Lab: Project Iteration

In this lab, we'll do whatever Adam and Clara tell us to do.

Update: View the presentation

(instructions/MVP Scoping at CUNY.pdf) ### Optional Lab Resources:

Get it RITE: Rapid Iterative Testing and Evaluation (RITE)

Week Nine

Homework

Write a call for programmers (think an advertisement for a freelance site or similar) for a project for which you would be the client. I will choose a representative set of these projects and, next week, will post them for teams and individuals to take on as their client project.

Your description of the project should be 350-500 words and should have three sections.

1. Description (What is the software that you need built? What is its purpose? Who is it for?)
2. Requirements (What do you see right now as essential to the software? Are there any limitations that your team of software designers will need to know about?)
3. About you or your organization (Why should a team of designers want to work on your project? Why is this software meaningful, important, or urgent? Why are you the one to ask that this software be made?)

Your request for software should reflect the skills we've been learning in the course. In practice this means it should be either a static website, a web app, an API, a command line program, or a Python module (a library for a task that can be imported). The project should ideally be extremely small in scope. If it's an API, it should serve up a very small data set and consist of only a few request types. If it's a static site, it should be no larger than five pages. If it's a module, it should do one simple task.

It's all Right if your post is longer than 500 words. Please upload your posting to GitHub and link me to the page where it is visible (preferred) or email it to me by Tuesday night.

Discussion: Planning final projects

Our final project will involve working with a client to determine requirements, negotiate tradeoffs in choosing a technological stack, and implement a prototype or MVP. This assignment is more focused on process than results, and will be graded not on the end product but on documentation of the steps you take to get there.

Lab: SQL

Structured Query Language (SQL) is a domain-specific language for storing and retrieving data in a database. We will use the command line interface to SQLite3, a relatively lightweight relational database engine frequently used in devices such as mobile phones, to store and retrieve structured data.

Optional Lab Resources:

Learn SQL the Hard Way
SQL Tutorial on W3Schools
DHRI Session on Databases

Week Ten

Homework

Discussion: Model View Controller (MVC) Paradigm

In this brief discussion, we'll talk about the model, view, controller (MVC)paradigm, in which different parts of the code base manage state, display information, and manage logic, respectively.

Lab: Prototype Web App with Flask

In this lab, we'll create a small website with some basic logic using Flask.

Optional Lab Resources

Creating APIs in Python and Flask on the Programming Historian (section on creating a small test API)
The Flask Mega-Tutorial
Prototype Twitter Clone in Flask

Week Eleven

Homework

Discussion: Final Projects and White Paper assignment

Lab: More SQL

In this lab, we'll continue our work on SQL using SQLite.

Optional Lab Resources

DHRI Introduction to Databases
Learn SQL the Hard Way (must be purchased)

Week Twelve

Discussion: Spring cleaning

In this discussion, we'll do a little spring cleaning, discussing the course assignments and checking in on final projects.

Final Project Deliverable
Blog post or white paper

Lab: Deploying to Production

In this lab, we'll talk about getting your work into other people's hands. We'll focus on the web, and discuss GitHub Pages as a quick way to get static sties online. We'll also talk about options for putting web applications into production, though we won't be able to cover that fully in class.

Week Twelve

** Spring Recess, no classes **

Week Thirteen

Discussion: Projects

In this discussion, we'll catch up with our blog posts and client projects as we move into the end of the semester.

Lab: Introspection

In this lab, we'll work with an API and use introspection—looking at objects in memory—to find out what's going on with data loaded into Python.

Lab:

Week Fourteen

Homework

Read *From Deficits to Possibilities: Mentoring Lessons from Plants on Cultivating Individual Growth through Environmental Assessment and Optimization*

Discussion: Mentorship

In our final discussion, we'll talk about building community, mentorship, and how to be a programmer in a humanities context... or a humanist in a programming context.