IS 597PR: Programming & Quality in Analytics
Course Syllabus - Fall 2022

Last modified: Aug 20, 2022

School of Information Sciences
University of Illinois at Urbana-Champaign

Instructor: Mr. John Weible, PMP, MS, Senior Lecturer. jweible@illinois.edu

Always check Canvas for office hours, announcements, assignments, and weekly course schedule details.

DESCRIPTION AND OBJECTIVES

This is an intermediate-level Python programming course involving a broad range of data structures, practical skills, concepts, best practices, and tools needed for developing & modifying software to solve moderately complex problems & to evaluate and improve code maintainability & reliability. These skills are relevant to all contexts of programming, but scenarios and assignments will include numerical data analysis, Monte Carlo simulation design, processing pipelines, using data sets drawn from scientific, historical, business, government, and other contexts. Introduces test-driven design, OOP design concepts, profiling & performance analysis, concurrent processing. The primary learning objectives are to increase quality of program code, breadth of problem-solving ability, and to develop deeper critical understanding of work in data analytics and its common flaws.

CLARIFICATION OF SCOPE

The main purpose of 597PR is to improve your computer programming abilities, moving you further toward professional-level coding. Thus, whenever you apply these skills (in any language) to data science or any other software field, you will be more capable and effective, including in a collaborative environment. Most of our problem scenarios will be representative of data analytics and we will use several software libraries commonly used for data science with Python. **But this course does not teach statistics or machine learning**. The iSchool has many other courses, expert faculty, and independent study opportunities through which you can learn much more of those fields in combination with 597PR.

PREREQUISITES

At least two prior semester-length programming courses or 1 year of experience with any general-purpose language(s) AND including some prior familiarity with Python fundamentals. IS 430 with 4-credit option AND some other programming course (e.g. IS597PY or IS455 or IS507) may be barely sufficient IF you work hard and study independently beyond the lecture materials. Most students in the course have taken quite a few programming courses before.

Primary Target Students: Students who have some prior experience and basic competence coding in at least one general-purpose object-oriented programming language (e.g. Java, Perl, PHP, C++, Ruby, Python, etc.), but not significant formal study of the field. Experienced and strong students coming to the iSchool with a BS degree in Computer Science or Software Engineering are not expected to benefit much from 597PR – such students *should* already have most or all these technical skills. However, after having taught this course for 10 semesters, I've discovered that even some of the students with undergraduate degrees in Computer Science or Engineering still have not learned some of the fundamentals in this course. Sometimes, students take the course to deepen their Python-specific knowledge. Further, even if you're a great coder already, there's still a good chance you'll develop better critical understanding of Data Analytics work which is the parallel focus of the course.

SUPPLEMENTARY TEXTBOOKS

Students with modest prior experience in Python may still succeed in this course without purchasing a textbook. The books listed here are recommended as supporting explanatory materials. Each contains examples and exercises for self-

study. These are NOT stocked by the campus bookstore but can be purchased online or may be available with limited access in the UI Library's O'Reilly collection.

Summerfield: "Programming in Python 3", 2nd Edition, by Mark Summerfield, ©2010. [Recommended – beginner through advanced level concepts of Python]
McKinney: "Python for Data Analysis", 2nd Edition, by Wes McKinney, ©2017. [Focused on Pandas & Numpy – Mostly intermediate and some advanced level. Unfortunately, it skips many fundamentals of good programming and can even encourage sloppy practices that are too common in analytics – we'll discuss this in the course.]
Matthes: "Python Crash Course: A Hands-On, Project-Based Introduction to Programming" 2nd Edition, by Eric Matthes [mostly beginner-level, for supplemental self-study]

Additional readings from other sources will be assigned to supplement these books but will be available at no cost.

COURSE CALENDAR

Refer to the Moodle course site. A Syllabus is not a list of assignments nor a schedule, it is an overview of course expectations and policies.

TYPICAL COURSE TOPICS (not in this order):

Programming Concepts:

- Students' previous programming experience varies a lot, and some will have only a little Python knowledge. We start with a fast summary of the language, with some translation of terms and syntax from other languages as needed.
- Reinforce all the fundamentals while steadily introducing more advanced concepts and selected best practices from software engineering – aspects of coding quality that are ignored or not even known by many "data analyst" wannabes.
- Expand study of data structures and libraries beyond our intro courses (e.g. sets, queues, network graphs, Numpy multi-dimensional arrays, pandas Data Frames & Series, etc.)
- Write functions including those that can accept a variable number of parameters
- Object oriented classes, operator overloading, encapsulation and custom accessor methods, static methods, and inheritance
- Functional programming concepts: first-class functions; lambda functions; recursion; closures
- Program performance analysis and improvement using profiling, selective compilation, and introduction to algorithmic complexity and performance measurements; Big-O/Big-Θ/Big-Ω notation
- Monte Carlo simulation techniques and analysis
- Intro to parallel programming (multi-threaded and multi-process coding)
- Designing and using multi-stage data pipelines

Software Development Practices & Tools:

- Program quality considerations and code reviews; Refactoring and progressive improvement
- Debugging and Tracing Tools
- Programming teamwork
- Test-Driven Development: Using & writing automated tests, the TDD process and continuous integration.
- Version control/source code mgmt.: using git and GitHub. Do checkout, commits, versioning, and merging; Compare distributed vs. centralized VCS tools for context.
- Code profiling and timing for performance and bottleneck analysis
- Full-featured Integrated Development Environments (IDE)
- Jupyter Notebooks

TECHNOLOGY REQUIREMENTS

Participation in IS 597PR will require you to bring your own laptop computer to class. It needs to be reasonably current hardware and operating system in properly functioning condition. You can use a Mac OS X, MS Windows, or Linux-based computer, as the software and concepts we'll be using generally apply to any of those operating systems. However,

iPads and most other pure tablet computers are not capable of running the programming tools we'll be using, even with a keyboard. If you have any doubts, please consult the instructor.

All the software we'll use in the class will either be free open source or commercial with free licenses for academic use. You will receive instruction and support for installation and use of the required software as part of the course.

COURSE GRADING

The purposes of grades include to incentivize and *quantify* a student's levels of: 1) good **efforts** toward **learning** and **practice**; 2) **evidence of mastery** of specific concepts and skills; 3) tangible, relevant **accomplishments** in the course. Often, grades are supplemented with written or verbal feedback statements to highlight strengths and clarify areas needing stronger efforts to succeed.

In this course, there are dozens of concepts and skills but they fall into a few broad categories. If the grading system is working ideally, then an "A" grade should result only if a student consistently shows good effort in various ways AND that *before the end of the semester*, the student has shown sufficient evidence of competence or mastery of most of the course concepts. Lesser grades should result from proportionately lesser efforts or less mastery.

How do we evaluate those components of grades? Several actions by students are indicators of these.

1. Participation: Frequently sharing your observations, questions, and/or answers during class sessions (verbally and/or via chat). This is the strongest single factor in student success. Students who do this are intrinsically rewarded by their faster learning, they are sought out as teammates, and they are easily recognized by the instructor for the ***efforts of learning, study, & practice***. This also will frequently give **direct *evidence of mastery*** of a particular concept. Broadly, "class participation" also can include helpful commentary through the weekly online forums, and a few designated in-class or critique/review activities that are turned in but not counted as normal homework assignments.

   The operational difficulty is for the instructor to keep track of what topic(s) of mastery each student shows, live, during class. To make this feasible with large class sizes, students will complete a simple ***self-assessment*** during each class session. This will supplement the instructor's and TA's observations and help students be more mindful of their focus, study, practice, and engagement, or the lack thereof.

2. On-the-spot partial code reviews: We've always done occasional code-reviews during and outside class time. This semester, we'll try to squeeze this into class more often for more scalable learning from the homework programming challenges and your collective ideas about them. You are encouraged to offer your own work as an example for review and to share insights about others' work. Also, you can demonstrate your understanding during office hours when asked certain questions or to explain code that you or others wrote.

3. Homework Evaluations: Consistently starting homework early and turning it in on time is important, *regardless of completeness*. We are using a newly-designed rubric for most assignments and for the whole course, that aims to achieve and balance all these:
   A. Many of the assignments will identify "core goals" everyone should aim to minimally complete each time, plus "stretch goal(s)" that are more complex or advanced in some way. All students should *try* to achieve some or all the stretch goals, but we're not expecting everyone to do so every time – especially students with less prior experience.
   B. Encourage ***strong & ethical efforts*** toward completing and solving the assignments given, regardless of program perfection or completeness. Such submissions will receive a "good effort" score, and will also be considered "complete" if they achieved all the "core goals".
   C. Try to recognize outstanding efforts & submissions when they occur but draw students away from the "100% points mentality". Outstanding will be those that also achieve the "stretch goals", have minor or no flaws, and that show efforts at applying all the course concepts up to that date (such as quality factors).

D. Reduce the TA time spent grading several hundred detailed programs each month, by simplifying the rubrics and reducing the detailed written feedback on each one.
E. Free up more TA & instructor time for more instructionally-beneficial work such as office-hours discussions, competence/mastery assessments, and emails with students.

4. The **opposite** of strong & ethical efforts is either doing nothing or plagiarism (which is *worse* than doing nothing and hence has worse repercussions). See the Academic Integrity section. Besides the direct sanctions, any student who ever plagiarizes also gives us significant reason to doubt whether any of the other submissions they made are truly their work.

5. Quizzes & one mid-semester exam: People learn at different rates and styles than each other and have different strengths. We will have an on-site exam during the second half of the semester during class. It provides a complementary way for students to directly demonstrate their level of mastery for many course concepts. Especially students who are reluctant to speak up in class or who avoid coming to office hours for discussions may find this exam an attractive opportunity to gain some of those mastery points they're sacrificing in the other ways. We also have several progress quizzes you may take open book, on your timeframe, at home.

6. The final project is expected to demonstrate good understanding and competence with all the relevant material taught in the course. Given the time span, multiple rounds of review and feedback, and comprehensive nature of these projects, they are evaluated with expectation of higher standards than the weekly assignments. It comprises 35% of the course grade.

NO COURSE INCOMPLETES

Do not expect to request an incomplete status for this course. If you are unable to keep up with the classwork even with moderate assistance, please notify the instructor and drop the course before the last date permitted. ***The instructor will only grant incompletes for justifiably dire, unavoidable, and documented reasons such as protracted hospitalization.***

CLASSROOM CONDUCT [some here doesn't apply to online class sections]

Class sessions will include lectures and other activities. Respectful, relevant, orderly discussion is encouraged and expected. Cell phones should be silent and ignored except for 2FA requirements & building pass. The use of computers in the classroom is expected for course-relevant purposes such as experimenting with code, accessing Moodle site, consulting on-line documentation, taking notes, and doing in-class exercises. However, unrelated activities such as texting, instant messaging, emailing, gaming, Facebook, WeChat, and other recreational web browsing are limited to outside the class or during designated break times. Even if you have mastered the material being discussed at the time, those activities will be distracting to other students and/or the instructor.

Food & Drink: food should be kept out of the auditorium (snack outside before or after class and at break time if you need to) and make sure all liquids are in spill-resistant containers.

The class period is long, so we will have at least one break during each session. If you need the restroom or have a personal crisis during class, quietly dismiss yourself when needed. Your instructor also understands that personal and family emergencies can occur.

ASSIGNMENTS

Assignments will be posted as we discuss the relevant subject matter in class and are typically due about 6 days thereafter (before the next class). Due dates for all assignments and quizzes will be shown in Moodle. All submissions must be uploaded to the Moodle space unless otherwise indicated (many will be submitted in GitHub instead).

If you have a serious health or personal crisis affecting your performance, please let the instructor and your academic advisor know as soon as possible.

CLASS PARTICIPATION

Enrollment includes the expectation of regular, on-time attendance. The students who actively engage in class will understand and retain much more of the material than those who are just passive or being distracted by unrelated activities. Also, the class is far more fun, interesting, and dynamic when students share their knowledge and questions in class. That's why this is part of the grading system.

This will be discussed in more detail during class. Expectations include asking and answering questions in class, actively contributing to group activities, aiding other students via the Moodle forums, and sharing other relevant and helpful information. Some of the assignments or activities will require sharing the completed work you produced with peers to facilitate code reviews. That type of code review activity (whether in-class or written) may be reflected in assignment or participation grades.

ASSISTANCE, STUDY GROUPS, AND INDIVIDUAL WORK

The instructor and TA can provide some assistance outside the classroom as described above. Your instructor also permits and encourages students to study together in person or have discussions through Moodle forums to help each other learn and puzzle through the concepts of the course and to help troubleshoot technical problems that can occur using the software. **But be very careful that your assistance does not go so far as blatantly sharing solutions to assignments or answers to quiz questions. In such situations, both the copying and the sharing students will be penalized unless you take extreme care to mark and cite ALL sections that are not completely original work. (See Academic Integrity section).** The course grading system was redesigned to make it even less tempting to consider cheating of any form. If applying your own mind to work is an unusual thought for you, it's a shame, but you can start practicing to think now.

Team development skills are required for some assignments. However, individual study and skills practice is also critical to developing complete understanding so you can apply these skills in later courses or non-academic computing work. All assignments for which group submissions are permitted will clearly say so in the instructions. For all group assignments, every group member is expected to contribute substantially to the work, shared as equally as possible. If a situation arises where a group member does not participate and contribute substantially, inform the instructor and TA so compensation can be made as necessary.

ACADEMIC INTEGRITY

Unfortunately, despite every caution, the instructor still finds it necessary to file FAIR reports and apply sanctions on some students every semester for plagiarism or other cheating. Do not risk it. Violations can even be discovered and sanctioned after the semester is over and you thought your grade was permanent. Privately inform the instructor of any situation where you suspect another student of a violation. Your disclosure will be kept anonymous.

Read and understand the academic integrity policy of the University of Illinois, http://admin.illinois.edu/policy/code/article1_part4_1-401.html. By submitting any course materials for grading, you certify that all work presented is your own and has been done by you independently, or as a member of a designated group.

You may not use any program code or writing from anyone and claim it as your own work **even if you typed it, altered it, or adapted it**. Just like when writing an academic term paper, you must cite all your sources! If you use any chunk of code (even just 2 non-trivial lines) from a book, website, friend, peer student, or other source, then insert comments in that section of code that *clearly* show the source whence it was derived, and the extent of work that's not original. This is not only for honesty—it is an important professional habit that will reward you later.

The consequences for plagiarism, quiz/exam cheating, or other forms of academic dishonesty are severe. Students who violate University standards of academic integrity are subject to disciplinary action, including a reduced grade, failure in the course, permanent report in academic records, and suspension or dismissal from the University. *Plagiarism on any assignment will result in at least a 1-letter grade reduction and the permanent record of infraction (via the FAIR system). Significant plagiarism on the final project at any stage (draft, presentation, or completion) will likely result in an F for the course, even if you had an A before that.*

Special notice about publishing program code: Some students are inclined to openly publish their programming assignment solutions online (most often in GitHub), misguidedly thinking this is how one gets a job. As *per the University's Student Code and this Syllabus, you may not at any time publish solutions for any course assignments* except when given explicit permission by the instructor. In this course, it will only be allowed for the final project. Doing so without permission undermines the course integrity and you will be reported for facilitating plagiarism. If you're not sure if your repositories are public (simple as it is), please ask to avoid a problem and penalty! To use such materials for any job interview or similar, it must be done in a way that is never public, such as by email it or granting the interviewing person(s) direct read access to a private Box folder or GitHub repository.

STATEMENT OF INCLUSION

http://www.inclusiveillinois.illinois.edu/chancellordivstmtswf.html#ValueStmt

As the state's premier public university, the University of Illinois at Urbana-Champaign's core mission is to serve the interests of the diverse people of the state of Illinois and beyond. The institution thus values inclusion and a pluralistic learning and research environment, one which we respect the varied perspectives and lived experiences of a diverse community and global workforce. We support diversity of worldviews, histories, and cultural knowledge across a range of social groups including race, ethnicity, gender identity, sexual orientation, abilities, economic class, religion, and their intersections.

DISABILITY STATEMENT

To obtain disability-related academic adjustments and/or auxiliary aids, students with disabilities must contact the course instructor and the Disability Resources and Educational Services (DRES) as soon as possible. To contact DRES you may visit 1207 S. Oak St., Champaign, call 333-4603 (V/TTY), or e-mail to disability@illinois.edu.