# CSE 312: Foundations of Computing II

Summer 2022

| | | | |
|---|---|---|---|
| **Instructor:** | Aleks Jovcic | **Time:** | MWF 12:00-1:00pm PST |
| **Email:** | ajovcic@cs.washington.edu | **Location:** | THO 119 |

**Course Resources:**

1. Course Website: https://courses.cs.washington.edu/courses/cse312/22su
2. Gradescope, EdStem, Calendar, and other materials linked from the website above.

**Announcements and Questions:** The class website will always have the most up-to-date information on the current schedule of topics to be covered and links to other class materials, including accompanying lecture notes, slides, etc. Announcements will be made via Edstem, and critical announcements will have an accompanying email. If you have any course content questions, you can ask them on Edstem. If you have any personal questions, please email me directly.

**TAs and Office Hours:** See website.

**Textbooks**

The only textbook we will be using is an ongoing textbook written by Alex Tsun from his earlier offering of 312. It is available for free on his website, linked on the course page.

Nonetheless, you may find the following further reading interesting:

- Larsen and Marx, *An Introduction to Mathematical Statistics* (5th edition). Prentice-Hall.
- Dimitri P. Bertsekas and John N. Tsitsiklis, *Introduction to Probability*, First Edition, Athena Scientific, 2000. Available online here.
- Sheldon Ross, *A First Course in Probability* (10th Ed.), Pearson Prentice Hall, 2018.

**Prerequisites:** CSE 311 and MATH 126. Here is a quick rundown of some of the mathematical tools we'll be using in this class: calculus (integration and differentiation), linear algebra (basic operations on vectors and matrices), an understanding of the basics of set theory (subsets, complements, unions, intersections, cardinality, etc.), and familiarity with basic proof techniques (including induction).

**Why is CSE 312 Important?**

While the initial foundations of computer science began in the world of discrete mathematics (after all, modern computers are digital in nature), recent years have seen a surge in the use of probability as a tool for the analysis and development of new algorithms and systems. As a result, it is becoming increasingly important for budding computer scientists to understand probability theory, both to provide new perspectives on existing ideas and to help further advance the field in new ways.

Probability is used in a number of contexts, including analyzing the likelihood that various events will happen, better understanding the performance of algorithms (which are increasingly making use of randomness), or modeling the behavior of systems that exist in asynchronous environments ruled by uncertainty (such as requests being made to a web server). Probability provides a rich set of tools for modeling such phenomena and allowing for precise mathematical statements to be made about the performance of an algorithm or a

system in such situations.

Furthermore, computers are increasingly often being used as data analysis tools to glean insights from the enormous amounts of data being gathered in a variety of fields; you've no doubt heard the phrase "big data" referring to this phenomenon. Probability theory and statistics are the foundational methods used for designing new algorithms to model such data, allowing, for example, a computer to make predictions about new or uncertain events. In fact, many of you have already been the users of such techniques. For example, most email systems now employ automated spam detection and filtering. Methods for being able to automatically infer whether or not an email message is spam are frequently rooted in probabilistic methods. Similarly, if you have ever seen online product recommendation (e.g., "customers who bought X are also likely to buy Y"), you've seen yet another application of probability in computer science. Even more subtly, answering detailed questions like how many buckets you should have in your a hash table or how many machines you should deploy in a data center (server farm) for an online application make use of probabilistic techniques to give precise formulations based on testable assumptions.

Our goal in this course is to build foundational skills and give you experience in the following areas:

1. **Understanding the combinatorial nature of problems**: Many real problems are based on understanding the multitude of possible outcomes that may occur, and determining which of those outcomes satisfy some criteria we care about. Such an understanding is important both for determining how likely an outcome is, but also for understanding what factors may affect the outcome (and which of those may be in our control).

2. **Working knowledge of probability theory and some of the key results in statistics**: Having a solid knowledge of probability theory and statistics is essential for computer scientists today. Such knowledge includes theoretical fundamentals as well as an appreciation for how that theory can be successfully applied in practice. We hope to impart both these concepts in this class.

3. **Appreciation and understanding of probabilistic statements**: In the world around us, probabilistic statements are often made, but are easily misunderstood. For example, when a candidate in an election is said to have a 53% likelihood of winning does this mean that the candidate is likely to get 53% of the vote, or that that if 100 elections were held today, the candidate would win 53% of them? Understanding the difference between these statements requires an understanding of the model in the underlying probabilistic analysis, as well as the context and implications of the real world.

4. **Applications in machine learning and theoretical computer science**: We are not studying probability theory simply for the joy of drawing summation symbols (okay, maybe some people are, but that's not what we're really targeting in this class), but rather because there are a wide variety of applications where probability and statistics allow us to solve problems that might otherwise be out of reach (or would be solved more poorly without the tools that probability and statistics can bring to bear). We'll look at examples of such applications throughout the class. For example, machine learning is a quickly growing subfield of artificial intelligence which has grown to impact many applications in computing. It focuses on analyzing large quantities of data to build models that can then be harnessed in real problems, such as filtering email, improving web search, understanding computer system performance, predicting financial markets, or analyzing DNA. Probability and statistics form the foundation of these systems. Another example application area is the use of randomized algorithms and probabilistic data structures. These usually have simpler and more elegant implementations than their deterministic counterparts, and have more efficient time and/or space complexity. We will be learning about some of these applications and you will have the opportunity to implement some of these algorithms.

**Tentative Course Outline:**

- Combinatorics (Weeks 1-2)
- Probability (Weeks 2-3)
- Random Variables (Weeks 4-5)
- Multiple Random Variables (Week 6)
- The Normal Random Variable (Week 7)
- Statistics (Week 8)

**Lectures:**
We will be holding live lectures in THO 210 on Mondays, Wednesdays, and Fridays, 12:00PM – 1:00PM Pacific Time. The lectures will be recorded, and you will be able to access those recordings within a few hours after class.

In addition, linked from the website are video recordings that Alex Tsun made that accompany the provided lecture notes. You may find those helpful. They cover the same material that we cover in class, though sometimes in class we will use different examples to illustrate the same concept.

**Grading Breakdown:**

| | |
|---|---|
| Problem Sets (7) | 60% |
| Final Pset | 20% |
| Review Summaries | 10% |
| Concept Checks | 10% |

**Problem Sets**

- There will be 7 problem sets. All of them will involve a written part and many of them will involve a coding problem and/or ask you to engage in non-technical analysis and consideration. You will be submitting your homeworks on Gradescope (detailed instructions on the problem set handouts).
- The coding you do on the problem sets will be done in Python. The implementation you do will provide you with a deeper understanding of how the theory we learn in this class is used in practice and should be a lot of fun. Note that we do not expect you to have any experience or knowledge of Python – we will provide you with tutorials and other kinds of help to get you started. The coding parts of the homeworks will be written in Python3, with no exceptions. This because the coding parts will be autograded. There are no hidden tests, and you'll have unlimited attempts. Whatever you see last on Gradescope for that section will be your grade. You will be able to write and test them on the edstem platform, which means that essentially no setup is required. A huge bonus of this class will be that you will come away with basic, working knowledge of Python (which you will undoubtedly use in the future, and definitely if you take CSE 446, the machine learning class).
- We strongly encourage you to type the written parts up using LaTeX. There are links to resources for learning LaTeX on the website. There you can also find If you take other classes that involve a fair amount of math (such as the machine learning class CSE 446) or plan to write research papers, you will need to typeset in LaTeX anyway. It is a very useful skill, so you may as well start now.
- You **must** show your work. We are not asking for fully watertight proofs, but as much explanation as you would need to explain to a fellow classmate who hadn't solved the problem before. Be concise. A correct answer with no work is worth less than a wrong answer with some work. Guidance will be given in class on how to answer a question.
- You **must** tag the question parts of your homework correctly on Gradescope. Failure to do so may result in deductions for each untagged question. Please check your submission by clicking each question, and making sure your solution appears there. We recommend starting each problem on a new page to keep this simple.

- Regrade requests are due on Gradescope within **one week** of grades being published, and will open up 24 hours after grades have been released.
- All problem sets (subject to change) must be completed **individually**. However, it is allowed and encouraged that you brainstorm and collaborate with others in coming up with solutions. If you do so, you must list all your collaborators at the top of each homework and write up your own final solution individually (**you may not copy word for word!)**

**Review Summaries:**

- Throughout the quarter, we will have three Review Summary assignments. These are intended to function similarly to a summative assessment such as a quiz or midterm. However, they will not be technical questions like the problem sets. Instead, you will be summarizing your understanding of the previous units in an open-ended format; you are free and encouraged to represent your learning in any way that helps you. The goal is that these assignments will require you to do the review and studying that you would for an exam, without the pressure of one. Review Summaries must be completed **individually.**

**Concept Checks**

- Associated with each lecture, there will be a "concept check" for you to take on Gradescope. This will consist of 4-8 questions that test your basic understanding of the concepts covered in lecture. Occasionally, the concept check will review something you should have learned in a previous class. The questions are intended to be straightforward to answer; each concept check should not require more than about 20-30 minutes.
- Each concept check will be available within an hour of the end of class and is due 30 minutes before the next lecture.
- You can submit your answers as many times as you want; we will only grade the final submission. Correct answers will reveal the answer explanation; all other answers will not, so you can keep trying until you see the answer explanation.
- Concept Checks can not be submitted late, but earning 90% in this category leads to 100% in the grade book.

**Late Policy:**

- Problem Sets and Review Summaries: You have 8 late days during the quarter, but can only use up to 2 late days on any one problem set or review summary. Please plan ahead as we will not be willing to add any additional no-penalty late days, except in emergencies.
- If you run out of late days, you may still turn in an assignment late by emailing your submission to the instructor, but may only earn up to 50% of the points.
- Concept Checks can not be submitted late, but earning 90% in this category leads to 100% in the grade book.
- If you have extenuating circumstances that interfere with any of the above, please get in touch with the course staff as soon as possible.
- Late days **cannot** be spent on the final pset or final review summary.

**Attendance Policy:** Regular attendance to lecture and section is strongly recommended. Moreover, I encourage you in the strongest possible terms to ask questions during lecture and sections (as well as in office hours and on the discussion board). That will make the class more fun for all and you will definitely learn more and have an easier time with the homework!

Keep in mind that this class is fast-paced, and the problem sets will be challenging. Most of the sections will be devoted to giving you practice on problems similar to those on the psets.

**Academic Integrity:** Lack of knowledge of the academic honesty policy is not a reasonable explanation for a violation. Each student is expected to do their own work on the problem sets in CSE 312. Students may discuss problem sets with each other as well as the course staff, with the following caveats:

- Do not take away any notes or screenshots from your discussions with others.
- After discussing with others, take a 30 minute break before writing up your solutions.
- Cite the names of all your collaborators somewhere on your homework.
- Write up your solutions entirely on your own.

Excessive collaboration (i.e., beyond discussing problem set questions) can result in honor code violations. Questions regarding acceptable collaboration should be directed to the class instructor prior to the collaboration. It is a violation of the honor code to copy problem set solutions from others, or to copy or derive them from solutions found online or in textbooks, previous instances of this course, or other courses covering the same topics (e.g., STAT 394/5 or probability courses at other schools). Copying of solutions from students who previously took this or a similar course is also a violation of the honor code. Finally, it is worth keeping in mind that you must be able to explain and/or re-derive anything that you submit.

Violations of the above or any other issue of academic integrity are taken very seriously, and may be referred to the University Disciplinary Board. Please refer to the Allen School's Academic Misconduct webpage for a detailed description of what is allowable and what is not.

**Accommodations:**

- **Disability Accomodation Policy**: See here for the current policy.
- **Religious Accommodation Policy**: See here for the current policy.

**Acknowledgements:** Syllabus wording largely influenced by Lisa Yan, Chris Piech, and Mehran Sahami from Stanford University's CS 109, as well as Anna Karlin and Alex Tsun's offerings of CSE 312.