1. **Prerequisites**

   Students are expected to have the following background:

   - Knowledge of basic computer science principles and skills, at a level sufficient to write a reasonably non-trivial computer program in Python/NumPy. (CS106A or CS106B, CS106X.)
   - Familiarity with probability theory. (CS 109, MATH151, or STATS 116)
   - Familiarity with multivariable calculus and linear algebra (relevant classes include, but not limited to MATH 51, MATH 104, MATH 113, CS 205, CME 100.) Stanford Math 51 course text can be found here.

2. **Honor Code**

   We strongly encourage students to form study groups. Students may discuss and work on homework problems in groups. However, each student must write down the solution independently, and without referring to written notes from the joint session. Each student must understand the solution well enough in order to reconstruct it by him/herself.

   It is an honor code violation to copy, refer to, or look at written or code solutions from a previous year, including but not limited to: official solutions from a previous year, solutions posted online, solutions you or someone else may have written up in a previous year, and solutions for related problems. In particular, you may not compare your solutions and code to other students' or to resources you can find online. If you discuss a problem with others or use an online resource for help, you must cite this in your submission.

   Furthermore, it is an honor code *violation to post your assignment solutions online*, such as on a public git repo. We run plagiarism-detection software on your code against past solutions as well as student submissions from previous years. Additionally, you may not screen share your Overleaf doc or code to other students (including during office hours). Please take the time to familiarize yourself with the Stanford Honor Code and the Stanford Honor Code as it pertains to CS courses. If you're ever unsure about whether something is an honor code violation, please create an Ed post or reach out to Sauren Khosla (Head TA), sauren@stanford.edu /John Cho (Course Manager) johncho@stanford.edu.

3. **Friday TA Lectures**

To review material from the prerequisites or to supplement the lecture material, additional lectures led by TAs will be held on **Fridays 3:00 PM - 4:20 PM in Gates B12**. Attendance at these lectures is optional, but encouraged.

4. **Optional Discussion Sections**

   There will also be optional weekly discussion sections led by TAs. These sessions are meant to be interactive and in a small, traditional classroom setting. They will largely involve working through problems that are similar to the homeworks. We will have one discussion section a week: **Wednesdays 6 - 7 PM at 200-030, Lane History Corner, Main Quad**.

5. **Syllabus and Course Materials**

   There is no required text for this course. The syllabus and notes are permanently available.

6. **Course Calendar**

   A course calendar with times for the lectures, office hours, Friday TA lectures, and discussions can be found on Canvas. All assignment deadlines and the midterm date can be found in the syllabus.

7. **Ed and Gradescope**

   We use Ed for Q&A and Gradescope for assignment submission.
   Ed and Gradescope access will be granted after enrollment to the class as we periodically synchronize with the official course roster. If you enrolled in the class but do not have access to Ed, it should come within two days. If it has been more than that, send Sauren Khosla (sauren@stanford.edu) an email.

8. **Grading**

   There will be four assignments, one midterm, and a final project. The assignments will contain written questions and questions that require some Python programming. The grading breakdown is as follows: assignments are collectively worth 40%, the midterm is worth 20%, and the final project is worth 40%. The assignments are weighted by their respective point values - for example, if **{p1, p2, p3, p4}** denotes the point values of each assignment, then HW1 is worth **p1 / (p1+p2+p3+p4) * 40%** of the total grade. This quarter's grading

basis is letter grade or CR/NC. Please make sure on Axess that you are enrolled with your desired grading basis.

We highly encourage students to answer each others' questions on Ed. To incentivize this, we will be giving 2% extra credit of the total course grade to the **ten** students with the most instructor-endorsed responses on Ed by the end of the quarter. We will also award 1 bonus point for homeworks that are typeset.

9. **Submitting Assignments**

   To limit access to the assignments to only enrolled students and avoid the solutions to show up online publicly, the assignments will only be posted on Ed (not the course website nor Canvas). Assignments will be submitted through Gradescope. You will receive an invite to Gradescope for CS229 Machine Learning Spring 2023. If you have not received an invite email after the first few days of class, first log in to Gradescope with your @stanford.edu email and see whether you find the course listed; if not please post a private message on Ed for us to add you. All homework assignments must be submitted individually. The final project for the course can be submitted as a group of up to three members by adding group-members to a single submission on Gradescope.

10. **Late Assignments**

    Each student will have a total of three free late (calendar) days to use for homeworks. Once these late days are exhausted, any assignments turned in late will be penalized 20% per late day. However, no assignment will be accepted more than three days after its due date. Each 24 hours or part thereof that a homework is late uses up one full late day. Please note that late days are applied individually. Please keep in mind that **late days cannot be applied to the final project report and poster deadlines** due to constraints in the grading deadline.

11. **Regrade Requests**

    Students will have **one week** to submit regrade requests on Gradescope for assignments starting from when the grades are released.

12. **Lecture Video Policy**

Lectures will be held in person in the NVIDIA Auditorium. All lectures this quarter will be recorded and posted on Canvas soon after the lecture is given. You can find the recordings on the course Canvas page. These recordings might be reused in other Stanford courses, viewed by other Stanford students, faculty, or staff, or used for other education and research purposes. Note that while the cameras are positioned with the intention of recording only the instructor, occasionally a part of your image or voice might be incidentally captured. If you have questions, please contact a member of the teaching team.

13. **Midterm Policy**

    We will have an in-person 3-hour timed open-notes exam on **Thursday**, **May 18 from 6:00 pm - 8:00 pm**. For SCPD/NDO students: You will be taking the exam remotely and need to nominate an exam monitor. Please refer to the SCPD handbook for information about the process. In order to better accommodate our SCPD students in different timezones, we allow 24 hour time window starting from the date/time of the main on-campus exam time for SCPD students to choose the time stamped 3 hour exam.Please reach out to the SCPD exams office (scpd-exams@stanford.edu) if you have any questions on the exam process for SCPD students.

    The teaching staff will provide more details on the exam once it is finalized.

14. **Office Hours and Queuestatus**

    The office hour schedule will be posted on the course calendar on Canvas. We will hold remote (via Zoom) and in-person group office hours this quarter. We will be using QueueStatus to determine priority order on Zoom (remote OH) only. In person, we'll be writing names on a board & keeping track of who arrives when to determine priority. After putting your name in the queue (can be found here), please watch for messages from the TAs on QueueStatus.

15. **Incomplete Requests from Previous Quarters**

    If you have an Incomplete from a previous quarter and you wish to complete the course this quarter, please contact John Cho (johncho@stanford.edu) at the beginning of the quarter to notify us that you would like to complete CS229 this quarter.

16. **Auditing**

    Please fill out this form here, and we will review all the audit requests and add you to the course's Canvas page. Please note that auditors do not get access to the Ed forum (partly because we often have limited TAs to answer questions online) and thus do not get access to the assignments. Likewise, auditors do not have access to Gradescope to submit assignments. Please contact John Cho at johncho@stanford.edu if you have any questions.

17. **Students with Documented Disabilities**

    We assume that all of us learn in different ways, and that the organization of the course must accommodate each student differently. We are committed to ensuring the full participation of all enrolled students in this class. If you need academic accommodation based on a disability, you should initiate the request with the Office of Accessible Education (OAE). The OAE will evaluate the request, recommend accommodations, and prepare a letter for faculty. Students should contact the OAE as soon as possible and at any rate in advance of assignment deadlines since timely notice is needed to coordinate accommodations. Students should also send their accommodation letter to the teaching staff by making a private post on Ed, as soon as possible. Please note that OAE accommodations do not apply to group projects.

# FAQ

1. **Difference between 3 and 4 units**
    a. The class can be taken with 3 or 4 units for undergraduates and graduate students. There is no difference in workload between them. We set it up this way mainly to give people more flexibility, and you are welcome to pick either. We generally encourage students to register for 4, but if you would rather do 3 for any reason (such as if you have a cap on their number of units), registering for 3 is fine too (you do not need to ask for approval).
2. **Is this the same class as the free machine learning class?**
    a. No, that is a different class, which is not good for Stanford academic credit. You can learn more about it at www.ml-class.org.
3. **When will solutions for homeworks be released?**

a. Solutions will be released after homeworks have been graded and around the same time as grades are published. For HW0, solutions will be released soon after the submission deadline.

4. **Can I take courses that overlap with CS229?**
    a. Yes. If you require an instructor's signature, please reach out to Prof. Tengyu Ma.
5. **Why am I seeing an out-dated webpage with information from previous quarters?**
    a. We try our best to keep the website up-to-date starting from a few days before the quarter starts. You might want to force reload the page and override local cache.
    On Mac, use Command + Shift + R .
    On Windows/Linux, use Ctrl + Shift + R .
6. **Can I use Google Colab instead of running code locally?**
    a. We discourage the use of Colab for two primary reasons: 1) it's challenging to use Conda environments in Colab, and this could cause complex and avoidable bugs, and 2) one of the intended takeaways of this course is to be able to run python scripts on your laptop (locally). Unless you are very, very familiar with Colab, we highly recommend developing locally (we also won't be providing assistance in office hours for Colab-related bugs).
7. **How I should ask for TAs to help me debug code:**
    a. Please note that the teaching staff will not debug code longer than 2-3 lines via Ed. Learning to debug is a critical skill for software programmers, and remote requests for help with code usually end up with the teaching staff giving you the answer rather than you learning how to find the answer.
    b. Moreover, since programming at the level of CS106A/B is a prerequisite for this course and the course's focus is on machine learning techniques rather than coding, the TAs are **discouraged** from helping you look at and debug large blocks of your code during the office hours. The TAs are also generally discouraged to help debug compilation errors.
    c. The best way to use office hours and ask TAs for coding questions would be
        i. You should come to office hours having done your own legwork and ruled out basic logical errors. Identify the place where the error is suspected to come from by doing ablation studies. (Please see below for some common debugging tips.)

    ii. During the office hours, you should articulate what your goals are and what you have observed in your experiments, what you think might be the problem, and what advice you need to move forward.
    iii. The TAs will mostly help you by **looking at and analyzing the outputs of your code** instead of looking at the original code. Typical advice that the TAs might give you would be to ask you to do more analytical or ablation studies about your code. For example, when you observe that your test error does not decrease as training for longer, the TAs might ask you to check if your training error decreases. If your training error does not decrease, then the TAs might ask to check if the gradient of your algorithm is implemented correctly.
d. Here are some common debugging strategies that might be useful (courtesy of CS221)
    i. Construct small test cases that you have worked through by hand and see if your code matches the manual solution.
    ii. Spend some time understanding exactly what the test cases are doing and what outputs they are expecting from your code.
    iii. If possible, write your codes in small chunks and test that each part is doing exactly what you expect.
    iv. PDB is the default python debugger. It is very helpful and allows you to set breakpoints. You can set a breakpoint with the following line: import pdb; pdb.set_trace() .
    v. Printing the state of your computation frequently can help you make sure that things are working as expected and can help you narrow down which portion of your code is causing the bug you are seeing, e.g. print("var1 has current value: {}".format(var)) .
e. Debugging tips for timeouts:
    i. Set operations in general are pretty slow, so if you have any see if you can do them in some other way.
    ii. Check if all loops / linear operations are necessary. For example, with searching through a list for a specific item, sometimes you can make that constant time by giving each item an ID (say 0, 1, 2, 3) and then using a dictionary as a cache (although sometimes you just have to live with the cost).
    iii. If you have a specific helper function you are calling a lot, see if there is anything in there you can optimize!
f. Other debugging tips
    i. If you do not know what type a variable is, use type(.) .

ii. If you are running into issues where “None” pops up, a function may not be returning what you are expecting.
iii. For indexing into lists: example_list[a: b] is INCLUSIVE for a but EXCLUSIVE for b.
iv. If a function has optional arguments, make sure you are feeding in the proper arguments in the proper places (very easy to mess up).
v. Since python 3.6, you can use f-strings for printing debug messages, rather than format.
vi. Because of broadcasting and other implicit operations, it's useful to assert shapes of np arrays (and tensors for deep learning) after each operation that can change the shape.