SI 106 Course Syllabus

Jump to Today

Term: Winter 2021 Professor Steve Oney (Links to an external site.)

Instructor:

GSIs: • Hrishi Rao • Pei-Yao Hung • Samarth Puri • Victoria Cope

IAs: • Alexis Phillips • Angela Young • Chase Goldman • Eric Andrews • Jessica Zhang • John Roselli • Katie Wolberg • Rebecca Parada • Samantha Russel • Samuel Winter • Sofia Levinson • Tara Moore

Watch Parties for the reading material (Mon 10-11:30am ET): https://umich.zoom.us/j/92012558196 (Links to an external site.)

Office Hours: Held online. See the full schedule. (Links to an external site.) to ask for assistance outside of office hours.

Support: Join the course Slack channel

Discussion Sections

Feedback: Leave feedback on a **discussion section** (Links to an external site.). **Contact:** To send a regrade request, fill out this form (Links to an external site.). For anything else, e-mail 106W21instructors@umich.edu

to an external site.)

- Instructor: Steve Oney

• **003 + 004: TR 10-11:30am** (Links to an external site.)

- GSI: Victoria Cope

Note: all times are ET

Course Information

• **005: TR 11:30am-1pm** (Links to an external site.)

- GSI: Hrishi Rao

• **006 + 007: TR 1-2:30pm** (Links to an external site.)

- GSI: Samarth Puri

• **008 + 009: TR 2:30-4pm (Links to an external site.)**

- GSI: Pei-Yao Hung

• **002: MW 11:30am-1pm (Links**

• Academic Integrity, Giving and Receiving Assistance

Unless otherwise specified in an assignment all submitted work must be your own, original work. Any excerpts, statements, or phrases from the work of others, including code snippets, must be clearly identified as a quotation, and a proper citation provided. Generally, you will be permitted to work with others on the weekly problem sets (more details below) with the exception of one final problem which you will be required to do entirely on your own.

Any violation of the School's policy on Academic and Professional Integrity (stated in the Master's and Doctoral Student Handbooks) will result in serious penalties, which might range from failing an assign-ment, to failing a course, to being expelled from the program. Violations of academic and professional integrity will be reported to UMSI Student Affairs. Consequences impacting assignment or course grades are determined by the faculty instructor; additional sanctions may be imposed by the Assistant Dean for Academic and Student Affairs.

Giving and Receiving Assistance

The first time you learn technical material it is often challenging. We are going to cover a wide range of topics in the course and we will move quickly between topics. Because it is my goal for you to succeed in the course, I encourage you to get help from anyone you like.

However, you are responsible for learning the material, and you should make sure that all of the assistance you are getting is focused on gaining knowledge, not just on getting through the assignments. If you receive too much help and/or fail to master the material, you will crash and burn later in the semester. The final submission of each homework exercise must be in your own words.

If you receive assistance on an assignment, please indicate the nature and the amount of assistance you received. If the assignment is computer code, add a comment indicating who helped you and how. Any excerpts from the work of others must be clearly identified as a quotation, and a proper citation provided (e.g., in the comments of the code if it is a code fragment you have borrowed). If you are a more advanced student and are willing to help other students, please feel free to do so. Just remember that your goal is to help teach the material to the student receiving the help.

As a rule of thumb, if you work through a problem with others, generating the same code on each of your screens, when you get to the end of the problem set, you should be able to do both of the following:

Explain how all of the code works to an instructor;

Throw away your answer and reproduce it on your own, without any assistance from other students, in less than an hour.

Each week there will also be a free-form question to demonstrate your understanding of the material on the problem set. Your answer will have to go beyond what was specified in the problem set in some way, by explaining some subtlety or extending the solutions in some way. For this question, you must work individually and not in groups.

It is acceptable for this class to ask for and provide help on an assignment via the Facebook Q&A site, including posting code snippets. Please don't post complete answers, though. If it seems like you've posted too much, one of the instructional staff will contact you to let you know, so don't worry about it. When in doubt, err on the side of helping your fellow students. When you post requests for help, the instructional staff will coach you, privately or publicly, on how to ask a question the right way. It takes some work to post good technical help questions and often, in the process, you will figure out the answer yourself. That's a good outcome. If you do the work to post good questions, you will find it's amazing how generous people will be with their time and expertise, both in this class and later in the wider world. We'll be looking at some examples of this on StackOverflow and other sites as the semester goes on.

After a few weeks, you will notice that some students are consistently providing good responses to others' posts in the FB group. Please do not contact those students privately via email or FB message to ask for private help. They are being generous enough by providing assistance in the public channel, where it does the most good.

You may find that a study group that starts out early in the semester as a mutual aid society turns more one

sided as the semester goes on. If you find that you're always the one giving help and never asking for it, or vice versa, please break up the study group and find others to work with. It will be to the long-term advantage of everyone involved—you can really fall behind in understanding the material in this course if you're getting too much help on problem sets week after week, and it can be hard to recover.

To reiterate, the collaboration policy is as follows. Collaboration in the class is allowed (and even encouraged) for problem sets – you can get help from anyone as long as it is clearly acknowledged. Collaboration or outside help is not allowed on exams or reading responses, though you will be allowed to use some materials that you bring with you to exams. Use of solutions from previous semesters is not allowed. The authorship of any assignments must be in your own style and done by you, even if you get help. Any significant help must be acknowledged in writing.

• Accommodations for Students with Disabilities

If you think you need an accommodation for a disability, please let the instructors know at your earliest convenience. Some aspects of this course, the assignments, the in-class activities, and the way we teach may be modified to facilitate your participation and progress. As soon as you make us aware of your needs, we can work with the Office of Services for Students with Disabilities (SSD) to help us determine appropriate accommodations. SSD (734-763-3000; ssd.umich.edu (Links to an external site.)) typically rec-ommends accommodations through a Verified Individualized Services and Accommodations (VISA) form. I will treat any information that you provide in as confidential a manner as possible.

• Course Topic Coverage

This page lists the topics that we will cover this semester and the depth of understanding that we expect by the end of the semester. Building this level of knowledge will take time and practice so do not expect mastery (or even competency) the first time you encounter a concept. Every concept is classified into one of four levels of understanding that we expect:

• **Awareness**: Know that this concept exists and how it fits within the larger picture.
• **Literacy**: You do not need to remember this concept in detail but if you are reminded of what it does (and how to use it), you should be able to.
• **Competency**: Ideally you have memorized how to use this concept and have a good idea how and when to use it but you might sometimes need a quick reminder about the details.
• **Mastery**: Knowing this concept and how to use it should be almost second nature. You should not need to look up how to use it.

Note that succeeding on an exam requires not only understanding a given concept/Python feature but also knowing how to use that concept/Python feature to solve a problem. Further, these concepts will not be tested individually; most problems will use multiple Python features.

Variables, Statements, and Expressions

• **Mastery** of:
  - What an expression, value, type, and output are (and the differences between them)
  - Basic arithmetic operators (like +, -, /, *)
  - The remainder operator (%) and how to use it to determine if a number is even or odd
  - Exponentiation (**)
  - How to properly call a function (and what it means when a function "returns" something) ○ Type conversion functions for the types we have covered so far (int(), float(), str(), list(), etc.) ○ Variables
    - What a valid variable name is
    - How to assign (and re-assign) a variable
    - How to reference a variable's value
  - What the print() function does and how to use it
  - What the input() function does and how to use it

- Order of operations when evaluating an expression with parentheses
- Writing single-line code comments

• **Competency** with:
- What hard-coding means (and how to avoid it)
- The order of function execution in complex calls such as: sub(square(3), square(1+1)) •

Literacy with:
- Integer division (//)

Debugging

• **Mastery** of:
- What syntax vs runtime vs semantic errors are
- How to use the test cases that we (your instructors) write to test your code

• **Competency** with:
- How to use an error message to help figure out what is wrong with your program

• **Awareness** of:
- Different runtime error types. You do **not** need to memorize all of the different kinds of runtime errors (NameError, ValueError, etc.)

Modules

• **Competency** with:
- What modules are
- How to import a module given its name and how to use it given a list of functions in that module •

Literacy with:
- the random module
  - You do **not** need to memorize the functions in the random module.
    - If you are asked to use it, we will point to documentation on how to use it
  - Note: it's still worth going through the exercises that use the random module to get an idea of how to use modules

• **Awareness** of:
- The Turtle module
  - You do **not** need to use the Turtle module in exams
    - However, going through the Turtle exercises might help you better understand some other concepts

Sequences (strings, lists, and tuples)

• **Mastery** of:
- Writing string and list literals
- Indexing sequences
- Sequence concatenation
- Common methods:
  - .split() - both with and without an argument

• **Competency** with:
- Creating tuples through literals
- Slicing sequences
- Sequence repetition
- Somewhat common methods:
  - .join()

• **Literacy** with:
- Less common methods:
  - .count()

        - .index()

Iteration and the accumulator pattern

- **Mastery** of:
    - What a for loop is and how/when to use it
        - How to use a for loop to loop over any kind of sequence (including when the accumulator variable is a list or string)
        - What an iterator variable is
    - The accumulator pattern and how/when to use it
- **Competency** with:
    - The range() function
        - The fact that range() does not produce a list
    - How to iterate through indices
- **Awareness** of:
    - Image processing
        - You do not need to know the image processing functions covered (but going through them might help you understand iteration better)

Booleans

- **Mastery** of:
    - Boolean literal expressions (there are only two)
    - How to use the six common comparison operators (==, !=, >, <, >=, <=)
    - The two common containment operators (in, not in)
- **Competency** with:
    - The three common logical operators (and, or, not)
- **Literacy** with:
    - Operator precedence

Conditionals

- **Mastery** of:
    - How and when to use conditional statements (if/elif/else)
    - Chained conditionals
    - The accumulator pattern with conditionals
- **Competency** with:
    - Nested conditionals

Transforming Sequences and Mutation

- **Mastery** of:
    - What mutation means
    - Which Python types are mutable
    - The "is" operator and how it is different from ==
        - Note: you will almost always want to use ==
    - Aliasing (the concept; not the name)
    - Common mutation methods:
        - .append()
            - How .append() is different from concatenation
- **Competency** with:
    - What None is (and the fact that most mutation methods return it)
    - The .format() method
- **Literacy** with:
    - Less common mutation methods/statements:

        - del
        - .insert()
        - .sort()
        - .reverse()
        - .remove()
    - Other methods:
        - .upper(), .lower(), .count(), .index(), .strip(), .replace()

Files

- **Mastery** of:
    - How to read and write a file
- **Competency** with:
    - Familiarity with finding a file in your filesystem
    - How to read content from a CSV-formatted file
    - How to write content to a CSV-formatted file
- **Awareness** of:
    - `with`
        - You do not need to know how to use `with` to read a file (but you can)

Dictionaries

- **Mastery** of:
    - How to create a dictionary using a literal expression
    - How to get the value associated with a key in a dictionary
    - How to mutate a dictionary
    - Important methods:
        - .keys()
    - How to iterate through a dictionary's keys and values (including using the accumulator pattern) •

Literacy with:

- When a dictionary is the best data type for a given problem
- Less important methods:
    - .values()
    - .items()

Dictionary Accumulation

- **Mastery** of:
    - How/when to use the accumulator pattern when the accumulator variable is a dictionary ○ Using the accumulator pattern on dictionary keys and values (for example, to find the key with the highest or lowest value)

Defining Functions

- **Mastery** of:
    - The syntax of defining functions
    - The return statement
        - The difference between print and return
- **Competency** with:
    - Scope
        - local vs global variables
- **Literacy** with:
    - Understanding why we define functions

Tuple Packing and Unpacking

- **Competency** with:

    - How (and why) we use tuple packing and unpacking

  Indefinite Iteration

- **Competency** with:
    - How (and when) to use a while loop
    - What an infinite loop is (and how to avoid them)
    - Control flow statements:
        - break
        - continue

  Advanced Functions

- **Competency** with:
    - How to define a function with default values
    - How to call a function with keyword parameters

  Sorting

- **Competency** with:
    - How to call the sorted() function
        - How to use the reverse and key parameters
- **Literacy** with:
    - The .sort() method, which works the same as sorted() but mutates

  Nested Data & Nested Iteration

- **Mastery** of:
    - What nested data is
- **Competency** with:
    - How to extract data from nested dictionaries/lists/tuples
        - When to use indexing (to get a specific item) vs iteration (to get every item)
- **Literacy** with:
    - The difference between deep and shallow copies of nested data
- **Awareness** of:
    - The copy module

  JSON

- **Mastery** of:
    - Why we use JSON
- **Competency** with:
    - How to read and write JSON-formatted strings in Python
    - How to use JSON.dumps() to print a nicely formatted object (with the optional indent parameter)

  Test Cases

- **Mastery** of:
    - What `assert` statements do and how to write them
- **Competency** with:
    - The purpose of test cases
    - How to test functions that return values
    - How to test functions that have side effects
- **Awareness** of:
    - What a good test case is

  Exceptions

- **Literacy** with:
    - What exceptions are and the control flow around exceptions
    - When and how to use try/except

Internet APIs

- **Competency** with:
  - How to use the requests module
  - How to get data from a public API
  - What an API key is
- **Literacy** with:
  - What a query string is and how to build it
  - How to read API documentation to figure out how to use a new API

More on Accumulation

- **Competency** with:
  - list comprehensions
- **Literacy** with:
  - map
  - filter
- **Awareness** of:
  - zip

- COVID Statement

  For the safety of all students, faculty, and staff on campus, it is important for each of us to be mindful of safety measures that have been put in place for our protection. By returning to campus, you have acknowledged your responsibility for protecting the collective health of our community. Your participation in this course on an in person basis is conditional upon your adherence to all safety measures mandated by the State of Michigan and the University, including maintaining physical distancing of six feet from others, and properly wearing a face covering in class. Other applicable safety measures may be described in the Wolverine Culture of Care (Links to an external site.) and the University's Face Covering Policy for COVID-19 (Links to an external site.). Your ability to participate in this course in-person as well as your grade may be impacted by failure to comply with campus safety measures. Individuals seeking to request an accommodation related to the face covering requirement under the Americans with Disabilities Act should contact the Office for Institutional Equity (Links to an external site.). If you are unable or unwilling to adhere to these safety measures while in a face-to-face class setting, you will be required to participate on a remote basis (if available) or to disenroll from the class. I also encourage you to review the Statement of Students Rights and Responsibilities (Links to an external site.), which includes a COVID-related Statement Addendum.

- Exams in 106

  What we (want to) test for

  We do our best to test your ability to apply what we learned rather than memorization (for example, how to use a for loop to solve a given problem rather than what a for loop is). Every programmer forgets syntactic and trivial details ("what is the difference between .find() and .index()). We do mark off points for minor syntactic errors, but we try to write all of our questions to test your understanding of concepts rather than memorization. The course coverage page gives an overview of the concepts and level of depth covered.

  You will be much better off if you:

  Study the concepts that we discuss in the class

  Write down the syntactic details that you might forget onto your two-sided page of notes (see below) If you do forget a syntactic detail while taking the exam and it's not on your page of notes, try looking through the exam to see if any other questions jog your memory. We sometimes try to include easily-forgotten syntactic details as part of provided code in the exam.

Accommodations

If you require accommodations for your exam, please let us know in advance and bring an SSD form (see the "Accommodations for students with disabilities" page).

Late Policy

We will not accept late exam submissions. We provide a short "submission time" cushion after exams (please see the exam description to know when the deadline is). The Canvas submission page will shut down at that time and you will not be able to submit.

• Grading

Grades are out of a maximum of 1,000 points:

points.png

280 8 problem sets (35 points per, highest 8 of 9 grades)
300 2 Midterm Exams (150 points each)
250 Final Exam
70 Final Project
100 Practice Tool
+5
(bonus) if more than 90% of the class fills out end of semester course evaluations

This class is not graded on a curve. The grade/point breakdown is as follows:

A+>= 990 B+>= 870 C+>= 770 D+ (NRC*)>= 650
A>= 940 B>= 840 C>= 740 D (NRC*)>= 600 F (NRC*)< 550
A->= 900 B->= 800 C->= 700 D- (NRC*)>= 550
*In accordance with University policy, students who earn D or E grades will receive a No Record Covid (NRC) on their transcript and may request that it is converted to a letter grade.

Grades are rounded. 899.50 points is an A-. 899.49 points is a B+.

Note that grades are not assigned on a curve. You can all earn As. So feel free to help your classmates learn and succeed in the course! Typically, the median grade has been B or B+.

Exam scores are generally lower than grades on other work. Except for the exams, with enough effort you should be able to get close to 100% of the points. If you get all 450 non-exam points and 81% on the exams, you can earn an A-. With just 64% on the exams you can earn a B-, if you earn all the non-exam points. Of course, doing the hard work to get all the non-exam points will usually lead you to doing better on the exams as well.

Late Policy

• Mindset

Note: Please also see the student health and wellbeing and page.

Establishing a **growth mindset** (a belief that you will get better with practice) is crucial in this course, for the rest of your courses, and pretty much anything you decide to do. Students with a growth mindset tend to be less stressed and better performers than those with a fixed mindset (a belief that ability is mostly innate). To illustrate the two different perspectives (inspired by this paper):

	Fixed Mindset (toxic, even for high performers)	Growth Mindset (our goal)
My goal is to...	demonstrate that I have an aptitude for programming.	learn how to program and improve my ability to program. Progress will be slow at times.
Failure indicates...	that I am not a good programmer.	a lack of effort, poor strategy, or that I need to go back and review.
When I don't understand something, I...	am afraid to ask others because it shows that I am not a good programmer.	do this on my own.
Asking for help and asking questions...	shows that I am not a good programmer.	helps me better learn new concepts, reinforces things that I already understand, and allows me to get a new perspective on a concept.
When I make an effort, I...	indicates that I am not good enough to get better at programming.	make use of the help resources, which the instructor put in place precisely because many students have the same problem.
When I meet difficulty, I...	spend less time practicing.	spend more time practicing.
After encountering difficulty...	my performance is impaired.	my performance is equal or improved.

You may be surprised to know that **procrastination is often the result of perfectionism.** Perfectionists don't want to risk doing an imperfect job and put off their work until it's so close to the deadline that their fear of missing the deadline is enough to overcome the fear of imperfection. By overcoming this, you can be less stressed and perform better.

	The Procrastinator	Our Goal
When a problem set is due on Sunday...	I start close to the deadline. I'm working on the problem set too late to	done early, even if it's incomplete and sloppy. I have time to improve it without getting stressed out.
The first draft of my problem set answers should be...	100% perfect.	I start early in the week and am not afraid of getting stuck or having a sloppy first draft. Throughout the

my own. I'm usually able to get it done (even if it's sloppy) by cramming but I don't learn much from doing the problems.

because I can connect what we learn in class with the problem set questions and improve my answers over time.

When there is an exam... When I have

I cram in the few days before the exam. I don't do as well as I could have because I didn't have time to practice.

I study for the exam but it's all review because I had a chance to practice all of the material

a question...

it's right before the deadline so I'm often not able to get help. Even if I can get help, I don't really have time to learn the material; I just learn enough to apply it to this problem and move on.

I'm not rushed. I ask over Slack or the next office hours. I have time to meaningfully reflect on the answer.

attend office hours so I'm mostly on

week, I build a meaningful understanding of the material

because I know I'm not where I should be.

I passively build connections with the material.

When I think about this class... I feel guilty and stressed

I get myself to start work by building up...

about imperfection.

consistent discipline.

enough stress to overcome my worries

just enough for the assessments (problem sets & exams) so I'm going to forget everything I learned in a few weeks.

After the semester is over...

I might earn an OK grade but I learned

I (usually) have earned a great grade and better retain what I learned.

Finally, your attention span is your greatest asset. When you can, put yourself in a mindset and situation where you can concentrate on your work. It will pay off.

- Official Course Description

  This course is an introduction to programming with a focus on applications in informatics. It covers the fundamental elements of a modern programming language and how to access data on the Internet. It also explores how humans and technology complement one another, including techniques used to coordinate groups of people working together on software development.

  Where this course fits in the curriculum

  This course is the first in a two-course sequence introducing students to programming and the culture of programming, with a focus on applications for people, by people, and about people. For people means applications for end-users or analysts, as opposed to back-end or infrastructure software. By and about people refers to processing data traces of people's actions and interactions.

- Python 3 Programming Coursera Specialization (Links to an external site.)
- Pandemic Zoom Participation Policy

  I am joining the live sessions from home (just like you, most likely!) If my network connection drops or my video

freezes, please don't give up on the meeting! Hang out for at least 5 minutes while the instructor reboots or switches to his cell phone for Internet connectivity. Thanks!

General rules for Zoom use in this class:
You must always use your UMich credentials/account to access this course. If you have another Zoom account, it will not work!
Here is information on how to log into your U-M Zoom account (Links to an external site.)
If you have already used your UM email to create a non-UM Zoom account, please see this page for information on how to migrate your account (Links to an external site.). (Note that this can take several hours, so please do it well before the first class meeting.)
Please only connect using a Zoom client, preferably on your laptop/desktop (better than phone or tablet). NEVER on a web browser. It does not work as well.
Please keep your Zoom client updated to get the newest features.
Be sure your display name in Wolverine Access actually represents your preferred display name (see "Preferred" name in "Campus Personal Information").
Have your camera on as much as possible during class meetings (but mute your microphone as much as possible, too) I strongly encourage you to keep your camera on for community-building purposes (but will not require it). Check out your background when you are on camera. Remove or cover items you don't want others to see
Please set up a profile photo for your account (for when your camera is off)
Pets are welcomed!
Test your audio and video -- use headphones and a microphone are probably the best thing. Please mute when not talking. However, it is OK to make mistakes! We will forgive each other for barking dogs, crying children, sudden doorbells, etc.
What to do if you are on a low-bandwidth connection?
Leave video off when you don't need it
Turn off HD video (in the Zoom app video settings)
When you screen share, only do so for as long as necessary
If possible, use collaborative (e.g., Google) documents rather than screen sharing
Mute your audio when not speaking (do this anyway!)
To improve your overall Zoom performance, consider asking others at your location to limit their high bandwidth activities while you need to be online for a course or required meeting.
Avoid running other data-intensive programs during Zoom meetings, such as streaming video or music or other web sites with dynamic content. (Why are you watching streaming video during class?)
Credit: Professor Sandvig (Michigan) (Links to an external site.) and Cornell

- Student Mental Health and Well-being

The University of Michigan is committed to advancing the mental health and wellbeing of its students, while acknowledging that a variety of issues, such as strained relationships, increased anxiety, alcohol/drug problems, and depression, directly impacts students' academic performance.

If you or someone you know is feeling overwhelmed, depressed, and/or in need of support, services are available. For help, contact Counseling and Psychological Services (CAPS) at (734) 764-8312 and https://caps.umich.edu (Links to an external site.)/ during and after hours, on weekends and holidays or through its counselors physically located in schools on both North and Central Campus. You may also consult University Health Service (UHS) at (732) 764-8320 and https://www.uhs.umich.edu/mentalhealthsvcs (Links to an external site.), or for alcohol or drug concerns, see http://www.uhs.umich.edu/aodresources (Links to an external site.).

For a more comprehensive listing of the broad range of mental health services available on campus, please visit: http://umich.edu/~mhealth/

Course Summary:

Date	Details	Due
Thu Jan 21, 2021	Assignment DS 01: Introduction & Course Overview	due by 11:59pm
Tue Jan 26, 2021	Assignment DS 02: Variables, Statements, and Expressions	due by 11:59pm
Thu Jan 28, 2021	Assignment DS 03: Debugging, Modules, and Turtle	due by 11:59pm
Mon Feb 1, 2021	Assignment Problem Set 1	due by 10am

	Assignment Problem Set 1 (1 student)	due by 11:59pm
Tue Feb 2, 2021	Assignment DS 04: Sequences	due by 11:59pm
Thu Feb 4, 2021	Assignment Problem Set 1 (4 students)	due by 10am
	Assignment DS 05: Iteration and the Accumulator Pattern	due by 11:59pm
Mon Feb 8, 2021	Assignment Problem Set 1 (3 students)	due by 10am
	Assignment Problem Set 2	due by 10am

Tue Feb 9, 2021	Assignment DS 06: Booleans, Conditionals, and Assertions	due by 11:59pm
Wed Feb 10, 2021	Assignment Problem Set 2 (1 student)	due by 10am
Thu Feb 11, 2021	Assignment DS 07: Transforming Sequences and Mutation	due by 11:59pm
Mon Feb 15, 2021	Assignment Problem Set 1 (1 student)	due by 10am
	Assignment Problem Set 2 (2 students)	due by 10am
	Assignment Problem Set 3	due by 10am

	Assignment Problem Set 3 (2 students)	due by 10pm
Tue Feb 16, 2021	Assignment DS 08: Files	due by 11:59pm
Thu Feb 18, 2021	Assignment Problem Set 3 (2 students)	due by 10am
	Assignment DS 09: Dictionaries	due by 11:59pm
Fri Feb 19, 2021	Assignment Problem Set 2 (1 student)	due by 5:47am
	Assignment Problem Set 2 (1 student)	due by 12pm

Mon Feb 22, 2021	Assignment Problem Set 1 (1 student)	due by 10am
	Assignment Problem Set 2 (1 student)	due by 10am
	Assignment Problem Set 3 (3 students)	due by 10am
Tue Feb 23, 2021	Assignment DS 10: Dictionary Accumulation	due by 11:59pm
Fri Feb 26, 2021	Assignment Midterm #1 Review	due by 3pm
Mon Mar 1, 2021	Assignment Problem Set 4	due by 10am

Assignment

Midterm Exam 1

due by 11:35am

Assignment

Midterm Exam 1

(5 students)

due by 11:40am

Assignment

Midterm Exam 1

(7 students)

due by 11:45am

Assignment

Midterm Exam 1

(2 students)

due by 11:50am

Assignment

Midterm Exam 1

(1 student)

due by 12:05pm

Assignment

Midterm Exam 1

(1 student)

due by 12:10pm

	Assignment Midterm Exam 1 (1 student)	due by 12:25pm
	Assignment Midterm Exam 1 (4 students)	due by 12:35pm
	Assignment Midterm Exam 1 (1 student)	due by 12:45pm
	Assignment Midterm Exam 1 (2 students)	due by 1:05pm
	Assignment Problem Set 4 (2 students)	due by 5pm
Thu Mar 4, 2021	Assignment DS 11: Defining Functions & Tuple Packing and Unpacking	due by 11:59pm

Fri Mar 5, 2021	Assignment Midterm Exam 1 (1 student)	due by 3pm
	Assignment Problem Set 1 (1 student)	due by 4pm
Mon Mar 8, 2021	Assignment Problem Set 5	due by 10am
	Assignment Problem Set 4 (1 student)	due by 12pm
Tue Mar 9, 2021	Assignment DS 12: Advanced Functions (keyword & optional parameters, lambda expressions)	due by 11:59pm
Wed Mar 10, 2021	Assignment Problem Set 5 (1 student)	due by 10am

Thu Mar 11, 2021	Assignment DS 13: More About Iteration (while loops)	due by 11:59pm
Mon Mar 15, 2021	Assignment Problem Set 6	due by 10am
Tue Mar 16, 2021	Assignment DS 14: Sorting	due by 11:59pm
Thu Mar 18, 2021	Assignment DS 15: Nested Data, Nested Iteration, and JSON	due by 11:59pm
Mon Mar 22, 2021	Assignment Problem Set 7	due by 10am
Thu Mar 25, 2021	Assignment DS 16: Internet APIs I	due by 11:59pm

Mon Mar 29, 2021	Assignment Problem Set 8	due by 10am
Tue Mar 30, 2021	Assignment DS 17: Internet APIs II	due by 11:59pm
Thu Apr 1, 2021	Assignment DS 18: Exceptions	due by 11:59pm
Fri Apr 2, 2021	Assignment Midterm #2 Review	due by 3pm
	Assignment Midterm Exam 2	due by 11:35am
Mon Apr 5, 2021	Assignment Midterm Exam 2 (6 students)	due by 12:35pm

	Assignment Midterm Exam 2 (2 students)	due by 1:05pm
Thu Apr 8, 2021	Assignment DS 19: Test Cases	due by 11:59pm
Tue Apr 13, 2021	Assignment DS 20: More on Accumulation: Map, Filter, List Comprehension, and Zip	due by 11:59pm
Thu Apr 15, 2021	Assignment DS 21: Jupyter Multimedia	due by 11:59pm
Mon Apr 19, 2021	Assignment Problem Set 9	due by 10am
Tue Apr 20, 2021	Assignment DS 22: Final Project Work Session	due by 11:59pm

Fri Apr 23, 2021	Assignment Final Exam Review	due by 3pm
Mon Apr 26, 2021	Assignment Final Project	due by 10am
Wed Apr 28, 2021	Assignment Practice Tool	due by 5pm
	Assignment Final Exam	due by 6:05pm
	Assignment Final Exam (6 students)	due by 7:35pm
	Assignment Final Exam (2 students)	due by 8:35pm

Assignment

Midterm Exam 1 - Adjusted Score

Assignment

Midterm Exam 2 - Adjusted Score