Course Introduction

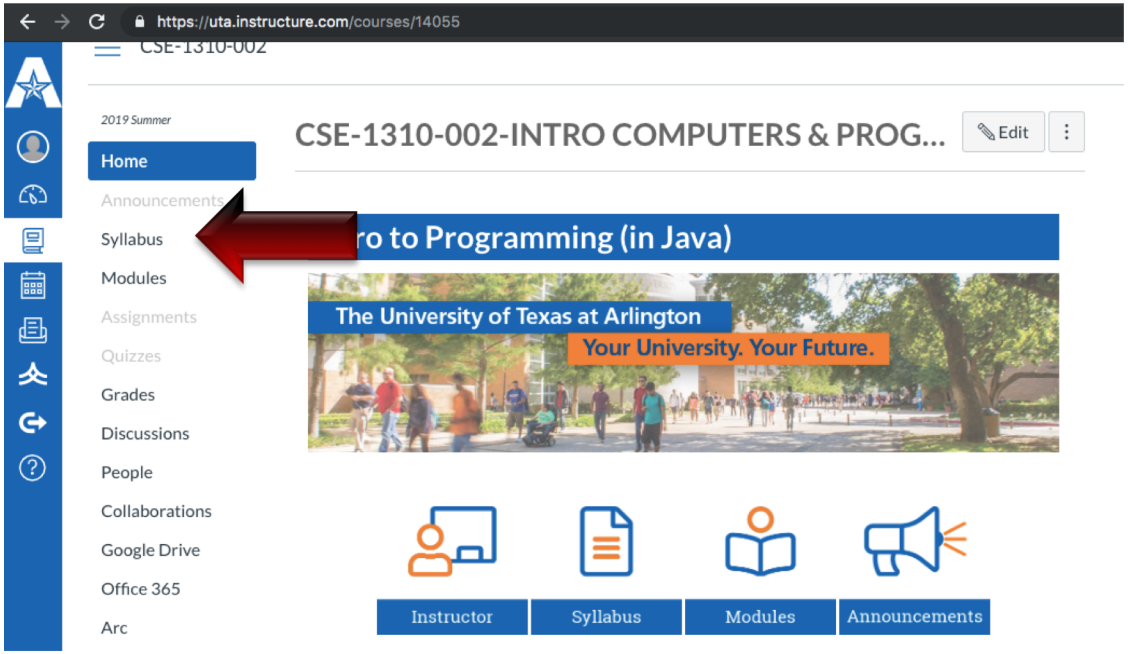
TOC

- Syllabus Highlights/Course Info
- How to Approach the Course

SYLLABUS HIGHLIGHTS/INFO

Syllabus Highlights

- Go over the syllabus on Canvas under the Syllabus tab (YOU SHOULD READ THE WHOLE THING YOURSELF):
  - Class info
  - Grading
- Schedule is on the Canvas homepage



Grading

- **DO NOT** ask me to email you your numerical grade at any point in the semester
  - You can calculate this yourself using the information on the previous slide
  - If you have any questions on how to calculate it, stop by my office hours to ask me how to do it-NO GRADES WILL BE RELEASED BY EMAIL
  - The syllabus also mentions what letter grade corresponds to a numerical grade
    - **DO NOT ASK ME TO CURVE-IF EVEN ONE PERSON ASKS ME, I WILL 100% NOT DO IT**

Time Expectations

- Expected: 10 hours per week outside of class.
  - Some people spend 5 hours/week and get an A.
  - Some people spend 20-30 hours/week and fail.
- Programming assignments will be time consuming.
- This is a class where:
  - Falling behind is easy.
  - Catching up is hard.
- Attention to detail will be crucial.
  - **Code that is 99% correct is 0% useful.**

Re-grading Policy

- Send the TA an email/go to their office hours asking about any grading questions (since they are the ones that do all the grading)
- If you have further questions about the grading or feel you do not agree with the TA grading, feel free to ask me
  - If you do this, I will send an email to the TA asking for a screenshot of the issue and will go from there

Assignments

- I don't give out HW solutions
  - If you have questions about HWs, feel free to as me or a TA about them
- I give **difficult HWs but make them a small percentage of your grade** so you can LEARN without harming your overall grade
  - This way, you can actually LEARN (the point of a university class)
  - They are completion grades
    - But remember that actually working on the HWs will be the key to doing well on the exams/quizzes

Regarding Submission Problems

- If, for whatever reason, you cannot submit on Canvas, then you can send us your assignment by e-mail.
- In that case, e-mail **(before the deadline)** your submission files to me and to the teaching assistant
  - If you e-mail us after the deadline, you still get the late penalty.
- Use your UTA e-mail, so that you can prove that you sent your message on time.
- Check with us ASAP to make sure we received your e-mail.
- You will still need to provide ASAP very convincing documentation that you really had problems with Canvas.

Quizzes

- Questions may ask you to write code, answer T/F questions and ask you to assign the correct vocabulary words to portions of code.
- Some questions may ask you to read code and:
  - Fix code
  - Predict what code will do
  - Write a short answer
  - Pick from multiple choice

Quizzes

- The material will be heavily based on the homework assignments and the class material (slides/class discussion/code)
- If you do your homework assignments and revise class material, the quizzes should not be too difficult
- NO MAKE-UP QUIZZES without documentation
  - You should also let me know ahead of time (including for religious holidays)

Syllabus

- The syllabus is posted on Canvas
- **You are responsible for reading fully and understanding what the syllabus says. If you have any questions or doubts, please ask me**
- At the end of the semester, when grades are posted, it is common to get complaints from students, saying that they were not aware of various aspects of the syllabus, that severely hurt their grade.
  - My only response will be to point them to this slide.

Getting Help

- Send me an email to set up an appointment (my email is on the syllabus
- DO NOT SEND ME EMAILS/MESSAGES ON CANVAS

Getting Help

- I try to answer emails within 24-48 hours
  - If you do not get a reply in that time frame, feel free to send a follow up email (I may have somehow missed the original email)
- If I am updating a grade or looking over a quiz, I will send you an email to confirm that I received it, but I may not actually do the required action until later on
  - I keep all emails in my inbox until I finish the task required by the email and will send an email when it is completed
  - If a long period of time goes by without an email signaling completion of the task, feel free to send a follow up email

Getting Help

- Additionally, feel free to email the TA with questions
  - There should be a TA lab set up-I will post more info about this when I get it
- **Do not expect responses to frantic queries in the last minutes before an assignment is due**.
  - I (and my TA) most likely won't be able to get to the email on time
- This strongly applies to debugging emails (see next page)

Debugging/Office Hours

- If you need your program debugged, send an email with your program and a short description of the problem
  - Due to the number of debugging questions I normally get, **I split the emails with the TA-you can send to either me or the TA**
    - Also, as mentioned previously there will be a TA lab with multiple TAs to help-I am just one person so only relying on me to debug may lead to delays
  - Do not just send your program and expect us to fix it for you-the reply will point you in the right direction
  - Include what you think may be causing the problem
  - SEND A SCREENSHOT OF THE ISSUE OCCURING WHEN YOU RUN THE PROGRAM

- **DO NOT** make a meeting for me to just debug a program
  - Debugging can take a considerable amount of time
    - I am essentially having to go through your whole code to find the error-this can take a lot of time
  - If you want to go over your code and how to fix it, send it to me first so I can look at it. We can then set up a meeting to go over it together

Computer Use

- If you do not have access to a personal computer, please let me know immediately
- Not having a computer is NOT an excuse to not complete assignments-we want all students to have equal opportunities and will make sure lab resources are available for you

Book

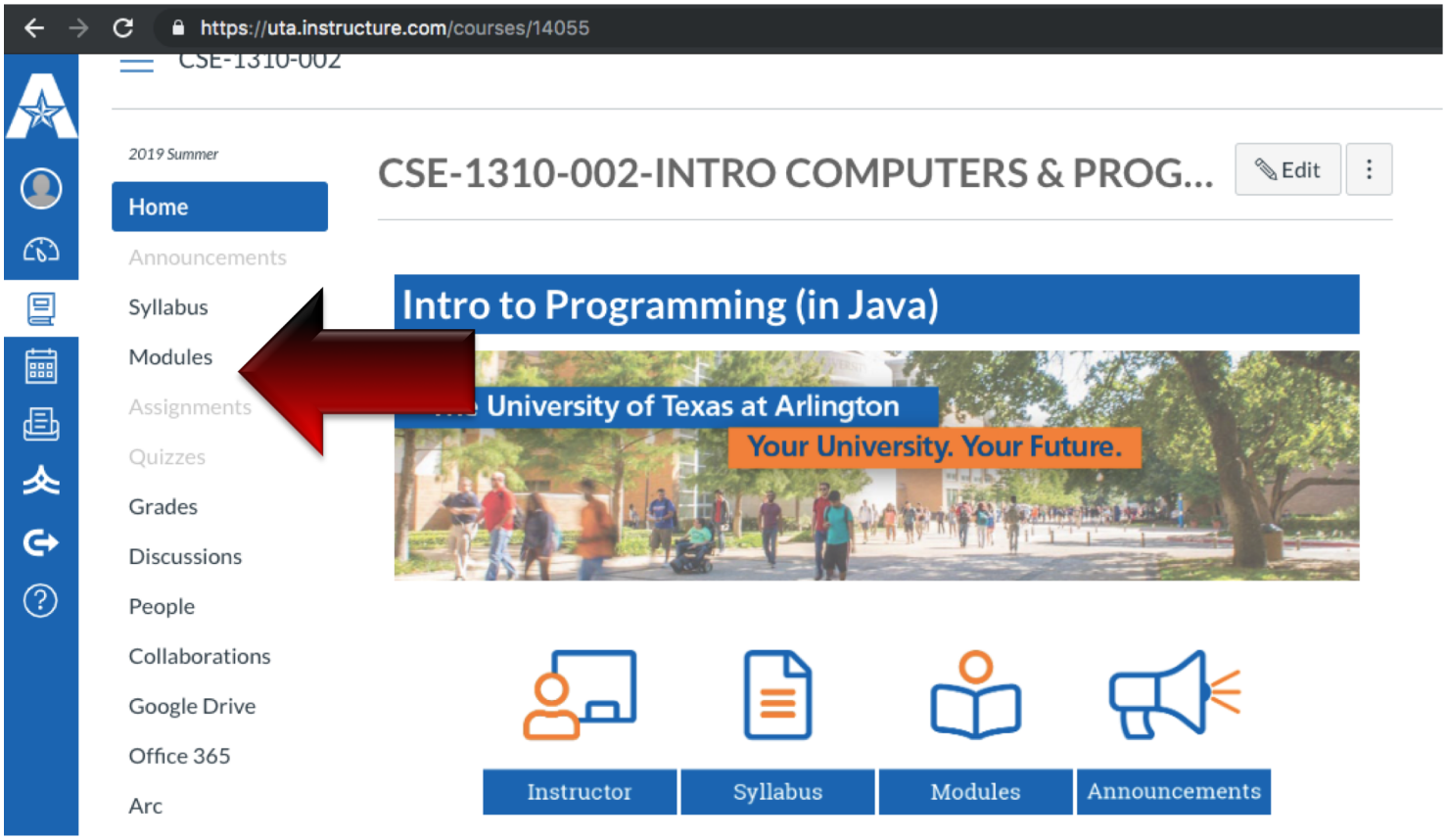
- The book is optional in this course
  - It is a good reference, but not necessary
- You will be tested on material from the slides, class discussions, code done in class and HWs
- I will also be showing you guys how to use online resources as a reference (since you will most likely be doing this in the future)
  - Knowing how to look up information is important

Resources

- One thing I emphasize in all my classes is that students should learn to use all available resources, including the internet
- Knowing how to locate information is a very important tool
- At some point, you will know multiple programming languages and it will be impossible to remember everything off the top of your head
  - As long as you understand the general concepts, finding the answer should be easy
    - For example, I can't remember every single function available in C, but I know how to find one I am looking for

Course Material

- Slides and code will be uploaded to the Modules tab



HOW TO APPROACH THE COURSE

Note

- I give some of my general tips, but I am aware everyone has their own technique
  - I’m just giving a starting point

During Class

- Find out what in-class techniques work best for you
  - Some students code along with me on a computer
  - Some students take notes
  - Some students simply watch and absorb

Study Tips

- Coding a little everyday is MUCH better than a lot occasionally
  - 20-30 minutes daily is better than 5 hours every 2 weeks
- YOU MUST ACTUALLY CODE TO LEARN-not just "review" code!
  - Passive (understanding) vs. Active (doing-producing code)
- Try to solve problems we do in class by yourself (only looking at the solution when you get stuck)

Study Tips

- I HIGHLY RECOMMEND WEBSITES LIKE:
  - Hackerrank
  - Kaggle (may be a bit more advanced)
  - Leetcode
  - etc

- These are amazing websites to start helping you to think like a computer scientist
- There are other sites-look them up!

Study Tips

- Since a main focus of this class is problem solving, it might be helpful to do math word problems
- The mental process you go through to solve these math word problems is the same process you go through when trying implement something in code
  - Think of it as exercise for your brain

Study Tips

- Learn how you learn!
  - Notice what works for you when you study
    - Some students like to code something multiple times
    - Some students prefer to always code something new
    - Some students like flashcards
    - See what works best for for you-it will make the rest of your computer science career much smoother
  - The one universal study tip for everyone is to CODE OFTEN AND FREQUENTLY (daily if possible)
    - Additionally, solving math word problems also helps you think more like a computer scientist

Study Tips

- Instead of taking notes ABOUT how something works, you may find it helpful to actually write a sample code and then put the notes about it in comments
  - It is more important to actually deal with code
  - Do not make the mistake of always worrying about the concept and avoiding code
    - Try doing the code and learning the concept that way

Study Tips

- Create your own problems to solve and code
  - Look around and see what interests you
  - Examples:
    - If you are interested in eating healthy during the semester, maybe make a program to count daily calories
    - If you are having trouble studying effectively, maybe make a program to help you organize your time
    - Etc.

- Most of the code I wrote for this class is a result of me creating my own little problems/scenarios and writing the code for it
  - The mental process you go through to recognize a scenario, solve it and code is pretty much what you will be doing your whole computer science career

Inevitable Feelings

- You will most likely feel one or more of the following at some point during the semester:
  - Will I ever get this material?
  - Why can't I understand this?
  - I will never get this! This is way too hard.
  - Why did I understand this yesterday but not today?
  - I am completely lost and don't even know how to get back on track. ☹

Inevitable Feelings

- Every language learner (whether you are learning Java or French) feels this way at the beginning
  - The trick is to keep going
  - I've never had a student that didn't give up not get it eventually
- As long as you **consistently** code, you will get there
  - **Actually write code**, don't just think about code
- I've heard multiple people say they "just woke up one day and got it"
  - This is actually due to consistently doing a little code every day-it all comes together

Approach

- I treat all programming classes as language immersion classes
  - This means I give you **a lot** of code in class
  - I obviously don't expect you to understand everything the first time you see it
- In each class session, I spend most of the time "translating" lines of code to you
  - Imagine that a complete program is a book and every line of code is sentence in that book
  - I'm "translating" these sentences to you

Approach

- I focus on:
  - Teaching you parts of the language
  - Syntax
  - A little “vocabulary” (mostly different methods/functions)
  - How to turn the real world around you into corresponding code
- My goal is for you to see and use enough code in different scenarios so you will be able to derive new code when you need

Approach

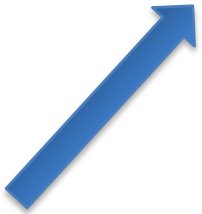
- Example in English:
  - If you understand the syntax and parts of speech, you know how to plug in

Nouns:	**Verbs:**
cat	to eat
dog	to drink
bird	to dance
…	…

Syntax example:

The _____ likes _____.

noun *verb*



We can have an infinite list of nouns and verbs and plug in and make a working sentence.

Approach

- Example in English:
  - If you understand the syntax and parts of speech, you know how to plug in

Nouns:	**Verbs:**
cat	to eat
dog	to drink
bird	to dance
…	…

✓ **Syntax example:**

The **cat** _____ likes **to eat** _____.

noun *verb*

X **Syntax example:**

The **to eat** _____ likes **cat** _____.

noun *verb*

We can have an infinite list of nouns and verbs and plug in and make a working sentence.

Approach

- I encourage you to type out class code yourselves to learn
  - Copying and pasting code doesn't help you
  - I know some of you will do this anyways, but just know it helps you more to type out the code by hand

Approach

- Passive vs. Active
  - Passive: reading the code, understanding it
  - Active: writing code, producing it
- You should definitely spend time reading code and understanding it, but to **actually produce code you must actually write code**
- Note that you will most likely be able to understand before you can produce
  - Its like talking to someone that understands English but can't really answer you back correctly

Approach

- Conclusion:
  - This is not a history or business class where you focus alone on learning the content and material
    - You will fail if you treat it as such
    - You can't *"cram"* the night before and memorize a bunch of material
      - You can't learn how to speak German in one night can you?
  - It's a:
    - **Language acquisition class** -learning the language
    - **Problem solving class** -how to think and solve a problem before expressing it in the language