Week 1: Lecture B

Python, Debugging, and VM Setup

Thursday, August 21, 2025

Reminders

- Be sure to join the course **Canvas** and **Piazza**
  - See links at top of course page
  - http://cs4440.eng.utah.edu
- Finish registering on **PollEverywhere**
  - Account must be `<yourUID>@utah.edu`
  - Location issues should be fixed
  - Sign in at https://pollev.com/cs4440
- Trouble accessing? See me after class!
  - Or email me at: snagy@cs.utah.edu

Reminders: Course Resources

Course website .... wiki, assignments, schedule, slides, office hours

Piazza .... questions, discussion, announcements

PollEverywhere .... lecture participation

Canvas .... quizzes, project submission, course gradebook

Instructor email (snagy@cs.utah.edu**)** .... administrative issues

Reminders: Weekly Quizzes

- First weekly **Lecture Quiz** released on **Canvas**
  - Submit by **11:59PM this Monday**
  - Late submissions not accepted

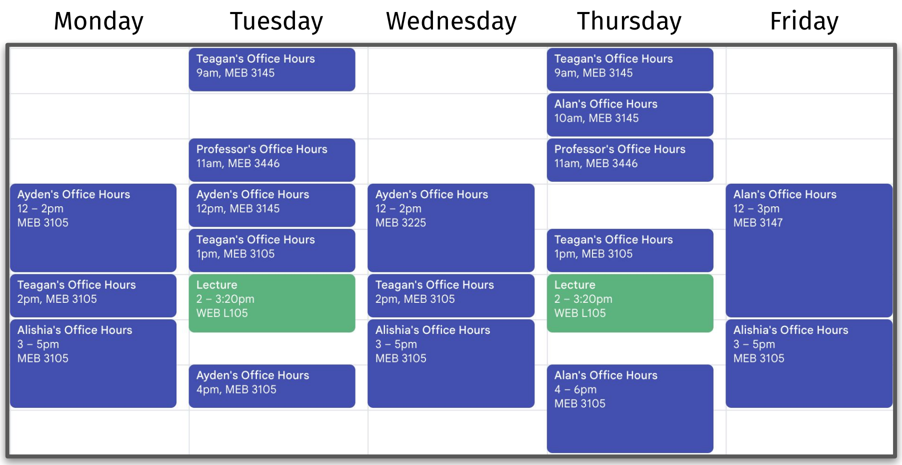
- Lecture quizzes released after Tuesday's lecture
  - Due the following Monday
  - Covers content from both Tuesday + Thursday lectures

Reminders: PollEverywhere

- **PollEverywhere:** check your UMail for an **account registration** email
  - We'll count today's attendance—let us know of any issues!
- Use your UID@utah.edu when participating
  - Should work automatically if you got the sign-up email

Reminders: Office Hours

- TA office hours (**24 total hours**)
  - First-come/first-serve via **TA Queue**
  - Help with programming projects
- Professor's office hours (**2 total**)
  - Help understanding lecture material
  - Administrative or grading issues
- Check the office hours calendar!
  - http://cs4440.eng.utah.edu
  - Cancellations announced via **Piazza**



Monday	Tuesday	Wednesday	Thursday	Friday
	Teagan's Office Hours 9am, MEB 3145		Teagan's Office Hours 9am, MEB 3145	
			Alan's Office Hours 10am, MEB 3145	
	Professor's Office Hours 11am, MEB 3446		Professor's Office Hours 11am, MEB 3446	
Ayden's Office Hours 12 – 2pm MEB 3105	Ayden's Office Hours 12pm, MEB 3145	Ayden's Office Hours 12 – 2pm MEB 3225		Alan's Office Hours 12 – 3pm MEB 3147
	Teagan's Office Hours 1pm, MEB 3105		Teagan's Office Hours 1pm, MEB 3105	
Teagan's Office Hours 2pm, MEB 3105	Lecture 2 – 3:20pm WEB L105	Teagan's Office Hours 2pm, MEB 3105	Lecture 2 – 3:20pm WEB L105	
Alishia's Office Hours 3 – 5pm MEB 3105		Alishia's Office Hours 3 – 5pm MEB 3105		Alishia's Office Hours 3 – 5pm MEB 3105
	Ayden's Office Hours 4pm, MEB 3105		Alan's Office Hours 4 – 6pm MEB 3105	

Note the rooms have changed!

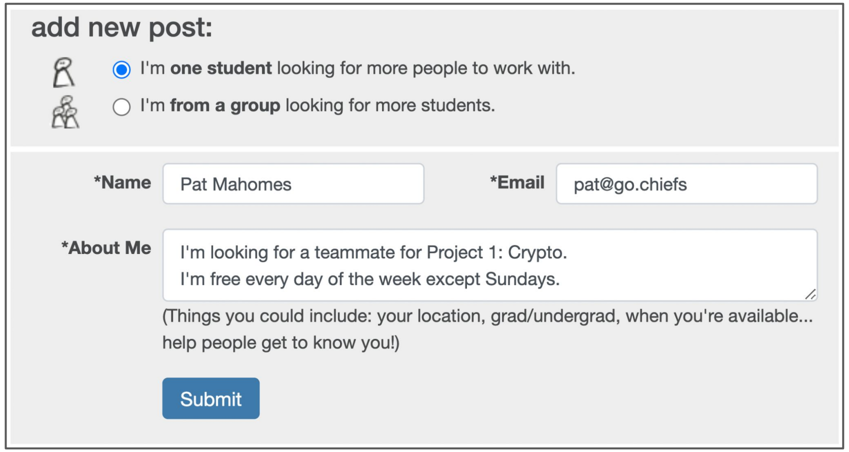
Reminders: Find a Teammate!

- Can work in **teams of up to two**
  - Find teammates on **Piazza**
  - Post on

    Search for Teammates! 12/21/22

- Why work with someone else?
  - Pair programming
  - Divide and conquer
  - Two sets of eyes to solve problems
  - Teaching others helps you learn more

- Yes, you are free to work solo…
  - But we encourage you to team up!



add new post:

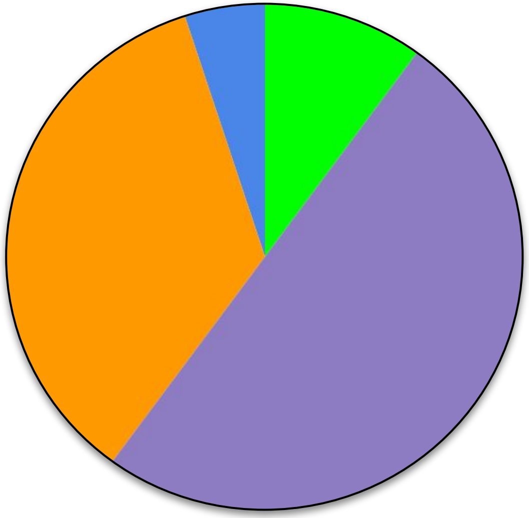
[● I'm **one student** looking for more people to work with.]

○ I'm **from a group** looking for more students.

*Name	Pat Mahomes
*Email	pat@go.chiefs
*About Me	I'm looking for a teammate for Project 1: Crypto. I'm free every day of the week except Sundays.

(Things you could include: your location, grad/undergrad, when you're available... help people get to know you!)

Reminders: Grading Breakdown



- 🟩 **10%** = weekly solo quizzes based on lectures
- 🟪 **50%** = four Programming Projects (**12.5%** each)
- 🟧 **35%** = Final Exam covering all course material
- 🟦 **5%** = participation during lecture poll exercises

Reminders: Collaboration Policy

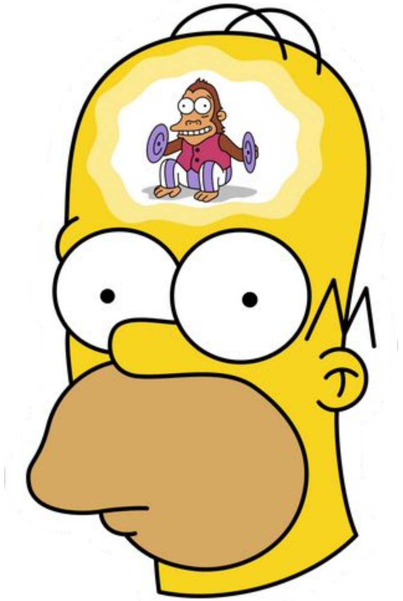
- We encourage you to help each other learn!
  - You may give or receive help on key **high-level concepts**
- However, **all code** must only be written by **you or your team**
- Cheating is when you give/receive an **unfair advantage**. Examples:
  - **Distributing your solutions** (e.g., to GitHub, Chegg, CourseHero) = **cheating**
  - **Copying code/solutions** (e.g., from GitHub, Google, another team) = **cheating**
  - **Copying code/solutions from AI tools** (e.g., CoPilot, GPT, Bard, etc.) = **cheating**
- Violations = misconduct sanctions. **Don't jeopardize your degree!**

Reminders: Participation

- **Lecture** participation via PollEverywhere:
  - **Three lecture absences allowed** at zero penalty
  - We'll track these internally—no need to notify us
  - Log-in as **your UMAIL** (e.g., `u8675309@utah.edu`)

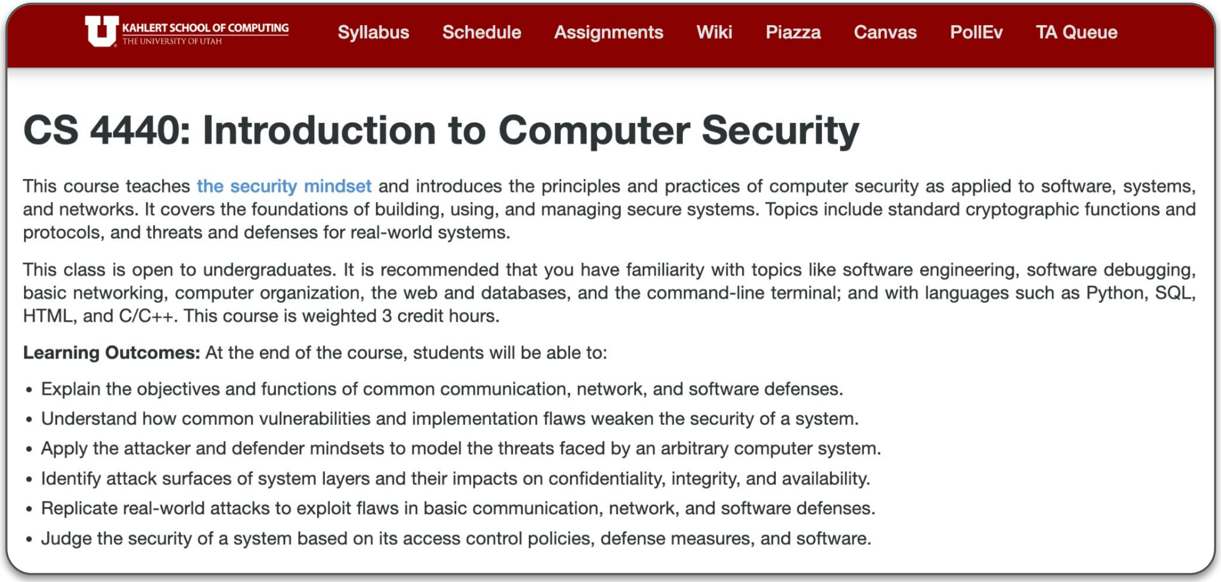
- **Online** participation on course Piazza:
  - Make intellectual contributions to help others learn
  - Collaboration policies apply—**don't share your code!**
  - **Top-10 answerers** will receive **5pts extra credit**

- How to **lose** points:
  - Frequently missing class, or not contributing online
  - Engaging in disruptive behavior or violating policies



Reminders: Course Website

- **Course website:** your go-to resource for all things CS 4440
  - http://cs4440.eng.utah.edu



CS 4440: Introduction to Computer Security

This course teaches the security mindset and introduces the principles and practices of computer security as applied to software, systems, and networks. It covers the foundations of building, using, and managing secure systems. Topics include standard cryptographic functions and protocols, and threats and defenses for real-world systems.

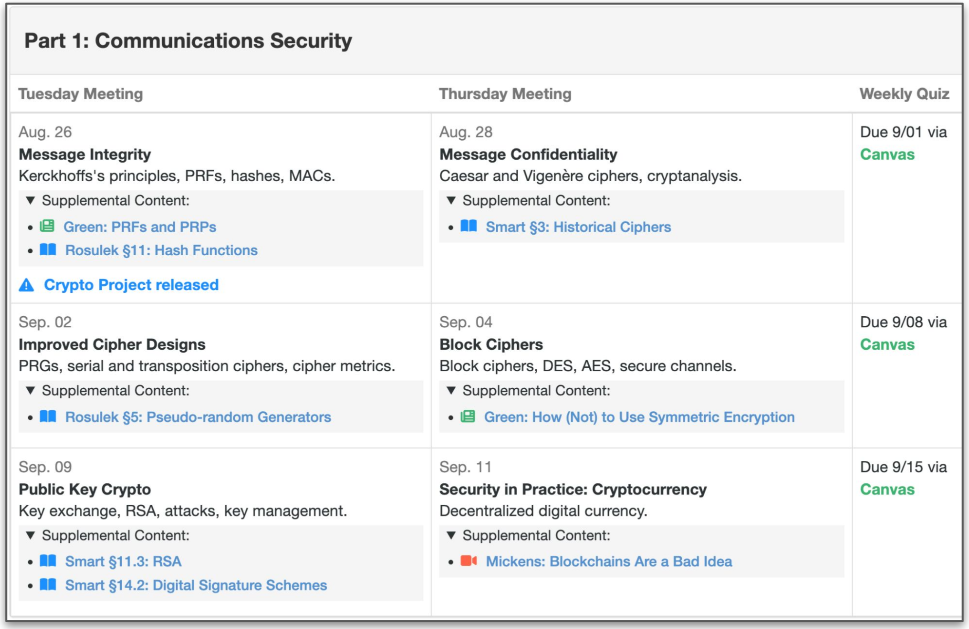
This class is open to undergraduates. It is recommended that you have familiarity with topics like software engineering, software debugging, basic networking, computer organization, the web and databases, and the command-line terminal; and with languages such as Python, SQL, HTML, and C/C++. This course is weighted 3 credit hours.

Learning Outcomes: At the end of the course, students will be able to:

- Explain the objectives and functions of common communication, network, and software defenses.
- Understand how common vulnerabilities and implementation flaws weaken the security of a system.
- Apply the attacker and defender mindsets to model the threats faced by an arbitrary computer system.
- Identify attack surfaces of system layers and their impacts on confidentiality, integrity, and availability.
- Replicate real-world attacks to exploit flaws in basic communication, network, and software defenses.
- Judge the security of a system based on its access control policies, defense measures, and software.

Reminders: Supplemental Content

- To further help you learn, we've provided **supplemental content** relevant to every lecture topic
  - Short blog posts
  - Free textbook chapters
  - Fun podcasts or videos
- **Totally optional**—not required
  - … though we do recommend them as additional resources to lectures!
- To access, click the drop-down "▸" button beside each lecture

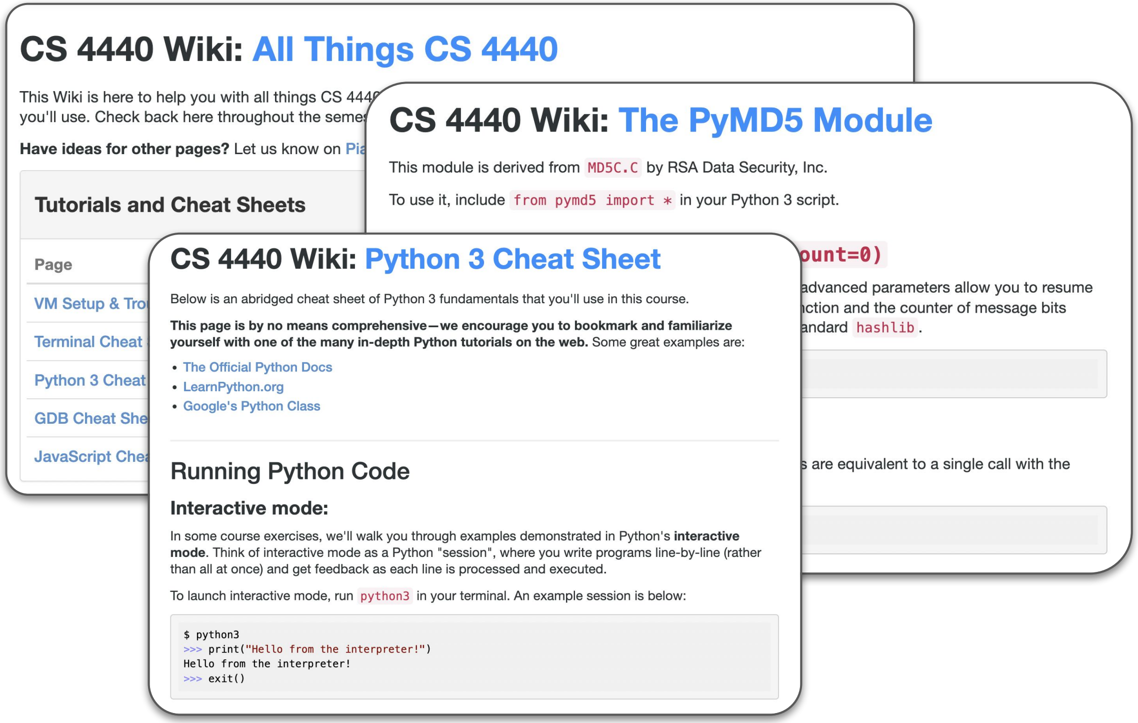


Part 1: Communications Security

Tuesday Meeting	Thursday Meeting	Weekly Quiz
Aug. 26 **Message Integrity** Kerckhoffs's principles, PRFs, hashes, MACs. ▼ Supplemental Content: • Green: PRFs and PRPs • Rosulek §11: Hash Functions ⚠ **Crypto Project released**	Aug. 28 **Message Confidentiality** Caesar and Vigenère ciphers, cryptanalysis. ▼ Supplemental Content: • Smart §3: Historical Ciphers	Due 9/01 via **Canvas**
Sep. 02 **Improved Cipher Designs** PRGs, serial and transposition ciphers, cipher metrics. ▼ Supplemental Content: • Rosulek §5: Pseudo-random Generators	Sep. 04 **Block Ciphers** Block ciphers, DES, AES, secure channels. ▼ Supplemental Content: • Green: How (Not) to Use Symmetric Encryption	Due 9/08 via **Canvas**
Sep. 09 **Public Key Crypto** Key exchange, RSA, attacks, key management. ▼ Supplemental Content: • Smart §11.3: RSA • Smart §14.2: Digital Signature Schemes	Sep. 11 **Security in Practice: Cryptocurrency** Decentralized digital currency. ▼ Supplemental Content: • Mickens: Blockchains Are a Bad Idea	Due 9/15 via **Canvas**

Reminders: Course Wiki

- Our aim is to lower the overall learning curve
- Resources to help you:
  - Tutorials
  - Cheat Sheets
  - Software documentation
- Many more resources added since last Fall



CS 4440 Wiki: All Things CS 4440

This Wiki is here to help you with all things CS 444
you'll use. Check back here throughout the semes

Have ideas for other pages? Let us know on Pia

Tutorials and Cheat Sheets

Page
VM Setup & Tro
Terminal Cheat
Python 3 Cheat
GDB Cheat She
JavaScript Chea

CS 4440 Wiki: The PyMD5 Module

This module is derived from `MD5C.C` by RSA Data Security, Inc.

To use it, include `from pymd5 import *` in your Python 3 script.

`ount=0)`

advanced parameters allow you to resume
nction and the counter of message bits
andard `hashlib`.

s are equivalent to a single call with the

CS 4440 Wiki: Python 3 Cheat Sheet

Below is an abridged cheat sheet of Python 3 fundamentals that you'll use in this course.

This page is by no means comprehensive—we encourage you to bookmark and familiarize yourself with one of the many in-depth Python tutorials on the web. Some great examples are:

- The Official Python Docs
- LearnPython.org
- Google's Python Class

Running Python Code

Interactive mode:

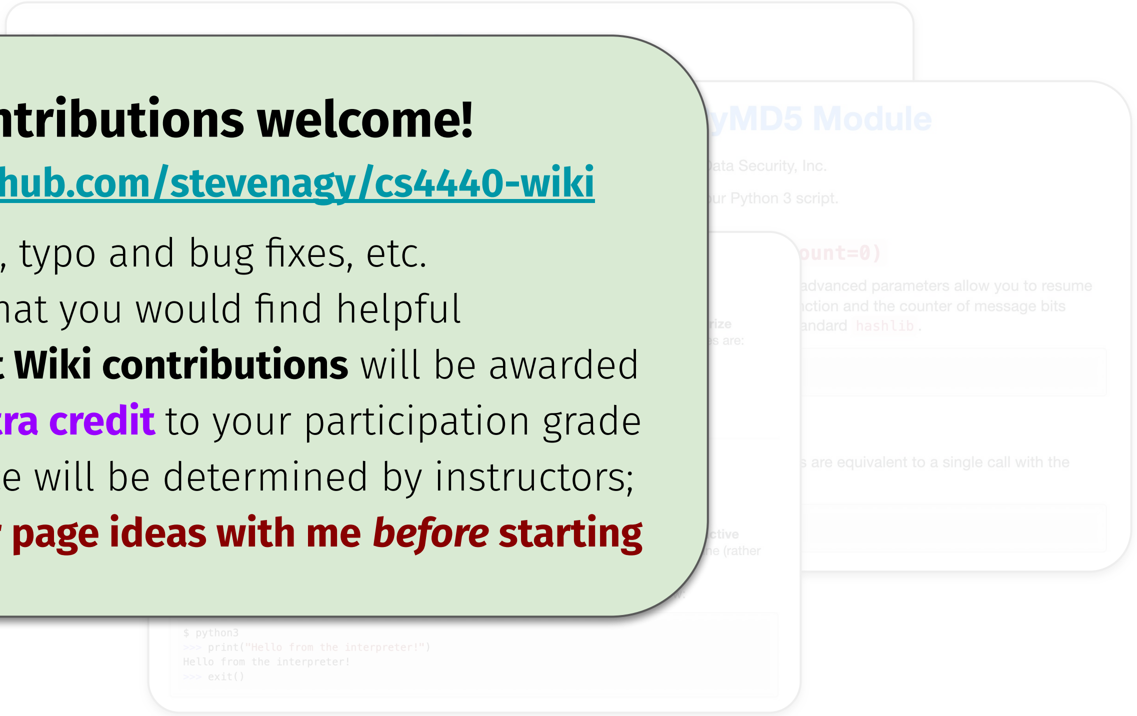
In some course exercises, we'll walk you through examples demonstrated in Python's **interactive mode**. Think of interactive mode as a Python "session", where you write programs line-by-line (rather than all at once) and get feedback as each line is processed and executed.

To launch interactive mode, run `python3` in your terminal. An example session is below:

```
$ python3
>>> print("Hello from the interpreter!")
Hello from the interpreter!
>>> exit()
```

Reminders: Course Wiki

- Our aim is to lower the overall learn
- Resources to
  - Tutorials
  - Cheat She
  - Software c
- Many more r added since



yMD5 Module

ata Security, Inc.

ur Python 3 script.

```
ount=0)
```

advanced parameters allow you to resume ction and the counter of message bits andard `hashlib`.

s are equivalent to a single call with the

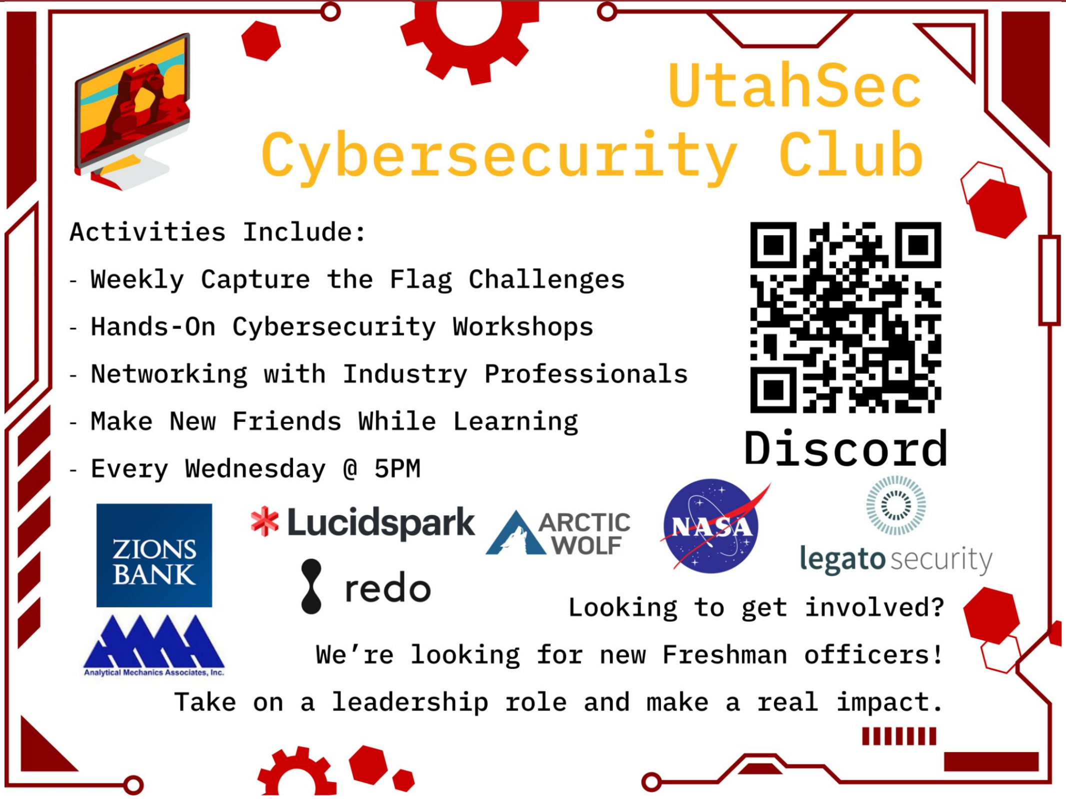
```
$ python3
>>> print("Hello from the interpreter!")
Hello from the interpreter!
>>> exit()
```

Contributions welcome!

https://github.com/stevenagy/cs4440-wiki

- Page ideas, typo and bug fixes, etc.
- Tutorials that you would find helpful
- **Significant Wiki contributions** will be awarded **1 point extra credit** to your participation grade
- Significance will be determined by instructors; must **clear page ideas with me *before* starting**

Reminders: The Utah Cybersecurity Club



UtahSec
Cybersecurity Club

Activities Include:

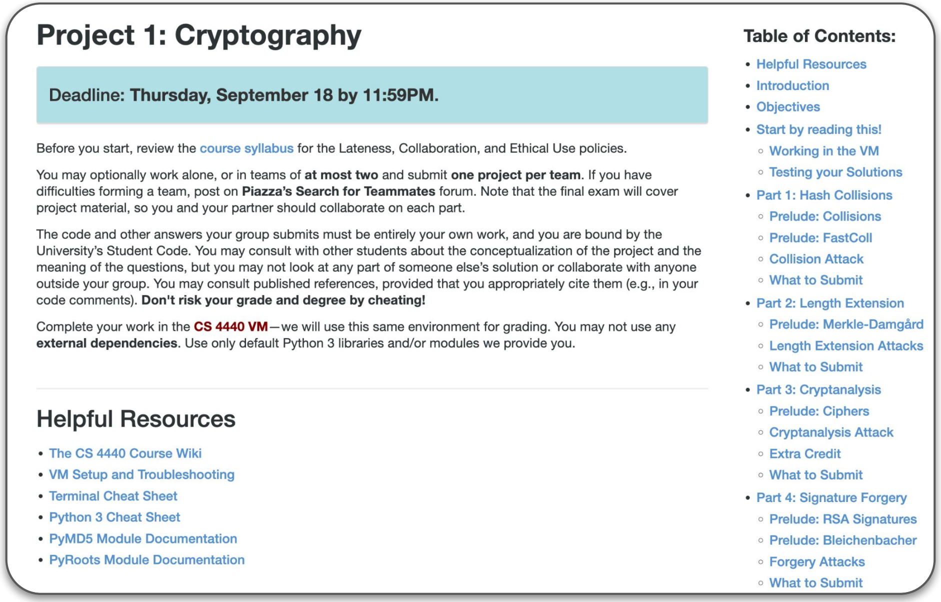
- Weekly Capture the Flag Challenges
- Hands-On Cybersecurity Workshops
- Networking with Industry Professionals
- Make New Friends While Learning
- Every Wednesday @ 5PM

Discord
ZIONS BANK
Lucidspark
ARCTIC WOLF
NASA
legato security
redo
Analytical Mechanics Associates, Inc.

Looking to get involved?
We're looking for new Freshman officers!
Take on a leadership role and make a real impact.

Announcements: Project 1

- **Project 1: Crypto** releasing on **Tuesday, August 26**
  - **Deadline:** Thursday, September 18th by 11:59PM



Project 1: Cryptography

Deadline: Thursday, September 18 by 11:59PM.

Before you start, review the course syllabus for the Lateness, Collaboration, and Ethical Use policies.

You may optionally work alone, or in teams of **at most two** and submit **one project per team**. If you have difficulties forming a team, post on **Piazza's Search for Teammates** forum. Note that the final exam will cover project material, so you and your partner should collaborate on each part.

The code and other answers your group submits must be entirely your own work, and you are bound by the University's Student Code. You may consult with other students about the conceptualization of the project and the meaning of the questions, but you may not look at any part of someone else's solution or collaborate with anyone outside your group. You may consult published references, provided that you appropriately cite them (e.g., in your code comments). **Don't risk your grade and degree by cheating!**

Complete your work in the **CS 4440 VM**—we will use this same environment for grading. You may not use any **external dependencies**. Use only default Python 3 libraries and/or modules we provide you.

Helpful Resources

- The CS 4440 Course Wiki
- VM Setup and Troubleshooting
- Terminal Cheat Sheet
- Python 3 Cheat Sheet
- PyMD5 Module Documentation
- PyRoots Module Documentation

Table of Contents:

Reminders: Project Lateness Policy

- Course staff constraints:
  - We want to return graded work promptly
  - Can’t discuss solutions until all work graded

- Project lateness policy:
  - **10% penalty** for being late up to **two days past deadline**
  - **Will not accept after 48 hours** past the original deadline
  - Extensions made only under **extraordinary** circumstances

- **Please start early!** It is your responsibility to...
  - Turn in assignments <u>ahead</u> of the deadline
  - Ensure your submissions <u>work</u> as intended

Questions?

Last time on CS 4440…

The Security Mindset
Modeling the Attacker
Assessing Risk
Secure Design

The Attacker

- Computer security studies how systems behave in the presence of an **adversary**
  - Independent / hobbyist hackers
  - "Script kiddies"
  - Cyber-criminal gangs
  - Nation-state government hackers
  - Disgruntled students (or professors)

- **Definition:** an intelligence that **actively tries to cause the system to misbehave**.

Thinking like an Attacker

- Look for the **weakest links**
  - What is easiest to attack
- Identify **assumptions** that the security depends on
  - Are any assumptions **false**?
  - Can you **render them false**?
- **Think outside the box!**
  - Don't be constrained by the system designer's worldview

Thinking as a Defender

- **Security Policy**
  - What resources are we protecting?
  - What properties are we enforcing?

Thinking as a Defender

- **Security Policy**
  - What resources are we protecting?
  - What properties are we enforcing?
- **Threat Model**
  - Who will attack us? Capabilities? Motivations?
  - What types of attacks must we try to prevent?

Thinking as a Defender

- **Security Policy**
  - What resources are we protecting?
  - What properties are we enforcing?
- **Threat Model**
  - Who will attack us? Capabilities? Motivations?
  - What types of attacks must we try to prevent?
- **Assessing Risk**
  - What are the system’s weaknesses?
  - How will successful attacks hurt us?

Thinking as a Defender

- **Security Policy**
  - What resources are we protecting?
  - What properties are we enforcing?
- **Threat Model**
  - Who will attack us? Capabilities? Motivations?
  - What types of attacks must we try to prevent?
- **Assessing Risk**
  - What are the system's weaknesses?
  - How will successful attacks hurt us?
- **Assessing Likelihood**
  - Countermeasures
  - Costs vs. benefits?
  - Technical vs. nontechnical?

Thinking as a Defender

- **Security Policy**
  - What resources are we protecting?
  - What properties are we enforcing?
- **Threat Model**
  - Who will attack us? Capabilities? Motivations?
  - What types of attacks must we try to prevent?
- **Assessing Risk**
  - What are the system's weaknesses?
  - How will successful attacks hurt us?
- **Assessing Likelihood**
  - Countermeasures
  - Costs vs. benefits?
  - Technical vs. nontechnical?

Rational paranoia:

Thinking **rigorously**, yet **realistically** about risk!

Security through... obscurity?

- **Common mistakes:**
  - Convincing yourself that a system is **already secure** in its current form
  - Convincing yourself a system is safe because attacker **won't know XYZ**
- **Better approach:**
  - **???**

Security through... obscurity?

- **Common mistakes:**
  - Convincing yourself that a system is **already secure** in its current form
  - Convincing yourself a system is safe because attacker **won't know XYZ**

- **Better approach:**
  - **Limit key assumptions** that security of your system depends upon
  - Identify **any components exposed** to attackers and their weaknesses
  - Assume **attacker knows everything** but a small bit of data (e.g., a key)

Rational Paranoia Exercises

Should you use a **strong password**?

- Assets?
- Adversaries?
- Risk assessment?
- Countermeasures?
- Costs/benefits?

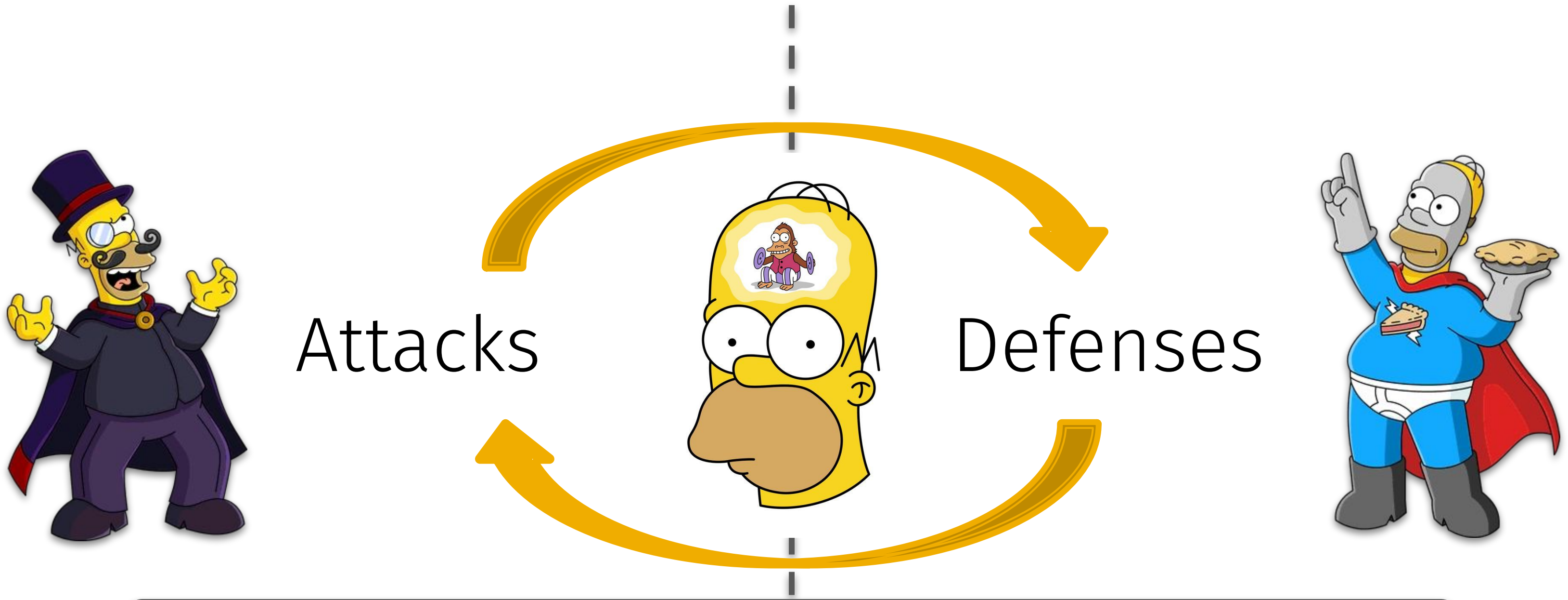
I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$WORDS.

Rational Paranoia Exercises

Using a **credit card** safely?

- Assets?
- Adversaries?
- Risk assessment?
- Countermeasures?
- Costs/benefits?

The Security Mindset



The Security Mindset: thinking as **both the attacker and defender**!

Questions?

This time on CS 4440…

Intro to Python
Debugging Code
Course VM Setup

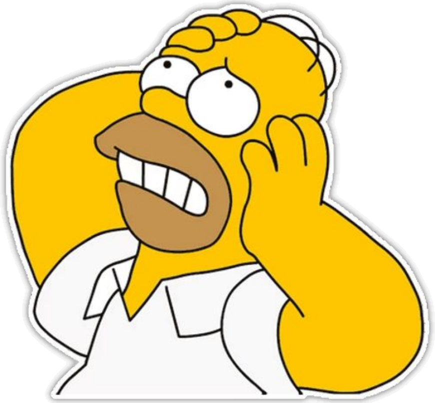
Languages and Tools in CS 4440

- Projects cover a few languages and tools:
  - **Project1:** Python 3
  - **Project2:** C/C++, x86, GDB
  - **Project3:** SQL, HTML, JavaScript
  - **Project4:** Python 3, Wireshark

Languages and Tools in CS 4440

- Projects cover a few languages and tools:
  - **Project1:** Python 3
  - **Project2:** C/C++, x86, GDB
  - **Project3:** SQL, HTML, JavaScript
  - **Project4:** Python 3, Wireshark

- This may seem daunting—but don't panic!



Languages and Tools in CS 4440

- Projects cover a few languages and tools:
  - **Project1:** Python 3
  - **Project2:** C/C++, x86, GDB
  - **Project3:** SQL, HTML, JavaScript
  - **Project4:** Python 3, Wireshark

- This may seem daunting—but don't panic!
  - Only using a **small subset** of their capabilities
  - We'll cover some basics in lecture as we go along
  - We'll post resources for you on the **CS 4440 Wiki**

Have you browsed CS 4440 Wiki yet?

Yes! 0%

No :(0%

An Intro to Python 3

Python 3

- Primary language for your Projects
  - Though expect to see some others too
- Characteristics:
  - High-level
  - Interpreted
  - Object Oriented
  - Dynamically Typed
  - Lots of indentation

python powered

```
print("Hello, world!")
```

Running Python Code

- Interactive mode
  - Launch Python 3 console
  - Enter code line-by-line
  - Executed line-by-line

```
$ python3
>>> print("Hello from the interpreter!")
Hello from the interpreter!
>>> exit()
```

Running Python Code

- Scripting mode
  - Edit your script (e.g., `MyScript.py`)
  - Then call the `python3` binary on it

```
$ cat MyScript.py
#!/usr/bin/python3
print("Hello from scripting mode!")
$ python3 MyScript.py
Hello from scripting mode!
```

Writing Scripts

- You’ll be writing relatively simple scripts
  - No need for an IDE
  - IDEs can/will break things

- Recommended text editors:
  - VIM
  - Nano
  - Emacs
  - FeatherPad
  - **Many others—pick one you like!**

Variables

- Can contain alphanumerical characters and some special characters

- Common conventions:
  - Variable names that start with lower-case letters
  - Class names beginning with a capital letter

- Some keywords are **reserved** (cannot be used as variable names)
  - Examples: `and`, `continue`, `break`
  - Python will complain if you use these

- **Dynamically typed:** a variable's **type** is derived from its **value**

Variables

- Types you'll likely see:
  - Integer (`int`)
  - Float (`float`)
  - String (`str`)
  - Boolean (`bool`)
  - Custom classes (e.g., `md5`)

Variables

- Types you'll likely see:
  - Integer (`int`)
  - Float (`float`)
  - String (`str`)
  - Boolean (`bool`)
  - Custom classes (e.g., `md5`)

- Variable assignment:
  - Assignment uses the "=" sign

```
>>> x = 5
>>> print(type(x))
<class 'int'>
```

Variables

- Types you'll likely see:
  - Integer (`int`)
  - Float (`float`)
  - String (`str`)
  - Boolean (`bool`)
  - Custom classes (e.g., `md5`)
- Variable assignment:
  - Assignment uses the "=" sign
  - Value changed? **So does type!**

```
>>> x = 5
>>> print(type(x))
<class 'int'>


>>> x = "cs4440"
>>> print(type(x))
<class 'str'>
```

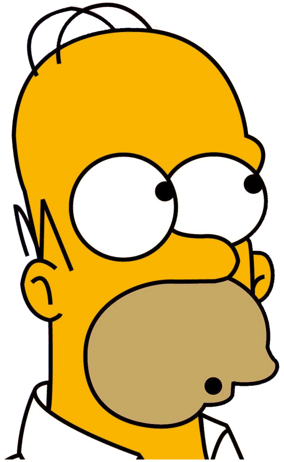
Variables

- Casting:
  - Pick a desired data type
  - “Wrap” your variable in it

```
>>> x = 5
>>> print(x, type(x))
5 <class 'int'>
```

Variables

- Casting:
  - Pick a desired data type
  - “Wrap” your variable in it
  - **Re-casting** will change type!



```
>>> x = 5
>>> print(x, type(x))
5 <class 'int'>

>>> x = float(x)
>>> print(x, type(x))
5.0 <class float>
```

Strings

- You will use **strings** in many exercises
  - Super flexible to use and manipulate
  - We'll cover some basic conventions

```
>>> x = "odoyle"
```

Strings

- You will use **strings** in many exercises
  - Super flexible to use and manipulate
  - We'll cover some basic conventions

- Basic string manipulation:
  - Length

```
>>> x = "odoyle"
>>> print(len(x))
6
```

Strings

- You will use **strings** in many exercises
  - Super flexible to use and manipulate
  - We'll cover some basic conventions
- Basic string manipulation:
  - Length
  - Appending

```
>>> x = "odoyle"
>>> print(len(x))
6


>>> print(x + "rules")
odoylerules
```

Strings

- You will use **strings** in many exercises
  - Super flexible to use and manipulate
  - We'll cover some basic conventions
- Basic string manipulation:
  - Length
  - Appending
  - Substrings

```
>>> x = "odoyle"
>>> print(len(x))
6

>>> print(x + "rules")
odoylerules

>>> print("odoy" in x)
True
```

Strings

- Other string manipulations:

```
>>> x = "cs4440:fa23"
```

Strings

- Other string manipulations:
  - Splitting by a delimiter

```
>>> x = "cs4440:fa23"
>>> print(x.split(':'))
['cs4440', 'fa23']
```

Strings

- Other string manipulations:
  - Splitting by a delimiter
  - Stripping characters

```
>>> x = "cs4440:fa23"
>>> print(x.split(':')
['cs4440', 'fa23']


>>> print(x.strip(':')
cs4440fa23
```

Strings

- Other string manipulations:
  - Splitting by a delimiter
  - Stripping characters
  - Repeating characters

```
>>> x = "cs4440:fa23"
>>> print(x.split(':')
['cs4440', 'fa23']


>>> print(x.strip(':')
cs4440fa23


>>> print('A'*10)
AAAAAAAAAA
```

Byte Strings

- Sometimes you will work with data as **bytes**
  - In Python, **byte strings** appear as `b'data'`
- Examples:
  - **Encoding** to a byte string

```
>>> x = "cs4440"
>>> x = x.encode(‘utf-8’))
>>> print(x, type(x))
b'cs4440' <class 'bytes'>
```

Byte Strings

- Sometimes you will work with data as **bytes**
  - In Python, **byte strings** appear as `b'data'`
- Examples:
  - **Encoding** to a byte string
  - **Decoding** a byte string

```
>>> x = "cs4440"
>>> x = x.encode(‘utf-8’))
>>> print(x, type(x))
b'cs4440' <class 'bytes'>

>>> y = x.decode(‘utf-8’))
>>> print(y, type(y))
cs4440 <class 'str'>
```

Byte Strings

- Sometimes you will work with data as **bytes**
  - In Python, **byte strings** appear as `b'data'`
- Examples:
  - **Encoding** to a byte string
  - **Decoding** a byte string
  - Must keep the same codec (e.g., `utf-8`)

```
>>> x = "cs4440"
>>> x = x.encode(‘utf-8’))
>>> print(x, type(x))
b'cs4440' <class 'bytes'>


>>> y = x.decode(‘utf-8’))
>>> print(y, type(y))
cs4440 <class 'str'>
```

Byte Strings

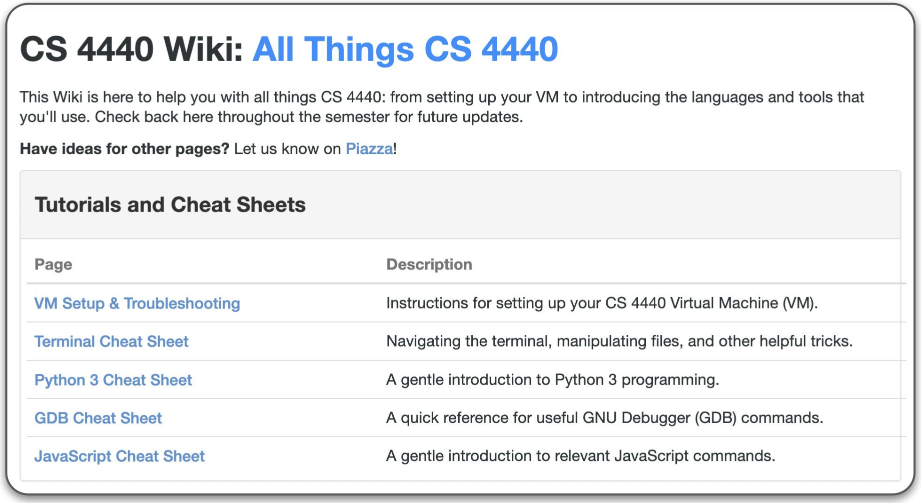
- Sometimes you will work with data as **bytes**
  - In Python, **byte strings** appear as `b'data'`
- Examples:
  - **Encoding** to a byte string
  - **Decoding** a byte string
  - Must keep the same codec (e.g., `utf-8`)
- Conceptually can be a little confusing
  - Functions `print()` and `type()` are your friends!

```
>>> x = "cs4440"
>>> x = x.encode('utf-8'))
>>> print(x, type(x))
b'cs4440' <class 'bytes'>


>>> y = x.decode('utf-8'))
>>> print(y, type(y))
cs4440 <class 'str'>
```

Other Key Concepts

- A few other concepts to review
  - Check these out in the **CS 4440 Wiki**



CS 4440 Wiki: All Things CS 4440

This Wiki is here to help you with all things CS 4440: from setting up your VM to introducing the languages and tools that you'll use. Check back here throughout the semester for future updates.

Have ideas for other pages? Let us know on Piazza!

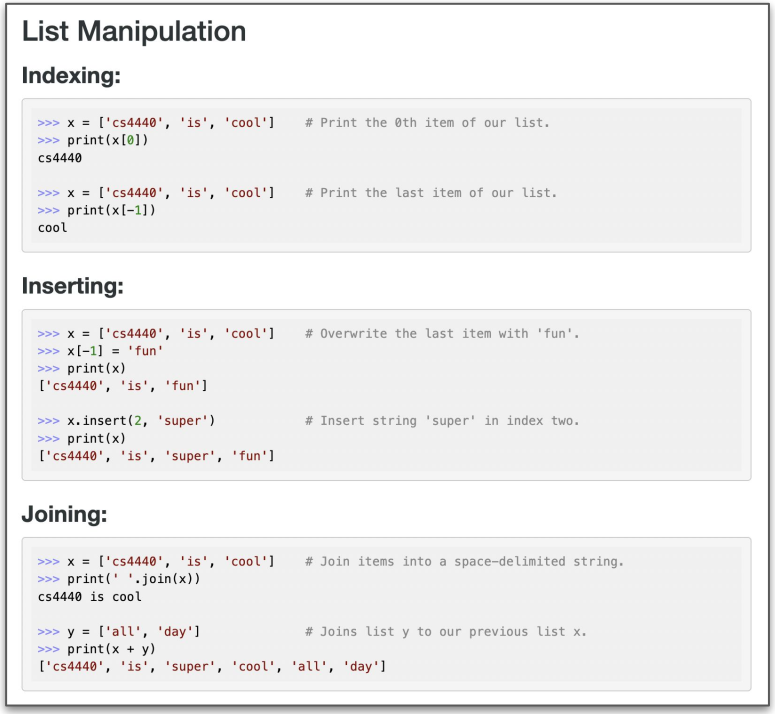
Tutorials and Cheat Sheets

Page	Description
VM Setup & Troubleshooting	Instructions for setting up your CS 4440 Virtual Machine (VM).
Terminal Cheat Sheet	Navigating the terminal, manipulating files, and other helpful tricks.
Python 3 Cheat Sheet	A gentle introduction to Python 3 programming.
GDB Cheat Sheet	A quick reference for useful GNU Debugger (GDB) commands.
JavaScript Cheat Sheet	A gentle introduction to relevant JavaScript commands.

Other Key Concepts

- A few other concepts to review
  - Check these out in the **CS 4440 Wiki**

- Lists
  - Appending
  - Prepending
  - Insert, Remove



List Manipulation

Indexing:

```
>>> x = ['cs4440', 'is', 'cool']    # Print the 0th item of our list.
>>> print(x[0])
cs4440

>>> x = ['cs4440', 'is', 'cool']    # Print the last item of our list.
>>> print(x[-1])
cool
```

Inserting:

```
>>> x = ['cs4440', 'is', 'cool']    # Overwrite the last item with 'fun'.
>>> x[-1] = 'fun'
>>> print(x)
['cs4440', 'is', 'fun']

>>> x.insert(2, 'super')            # Insert string 'super' in index two.
>>> print(x)
['cs4440', 'is', 'super', 'fun']
```

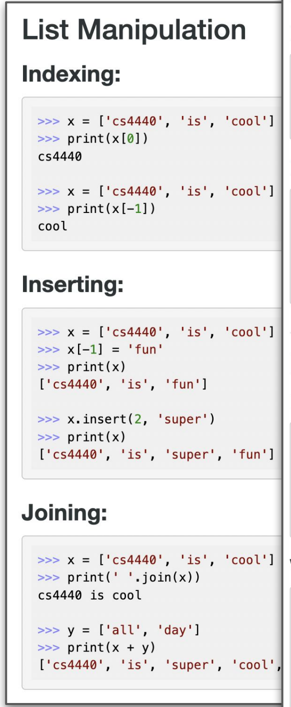
Joining:

```
>>> x = ['cs4440', 'is', 'cool']    # Join items into a space-delimited string.
>>> print(' '.join(x))
cs4440 is cool

>>> y = ['all', 'day']              # Joins list y to our previous list x.
>>> print(x + y)
['cs4440', 'is', 'super', 'cool', 'all', 'day']
```

Other Key Concepts

- A few other concepts to review
  - Check these out in the **CS 4440 Wiki**
- Lists
  - Appending
  - Prepending
  - Insert, Remove
- Control Flow
  - Loops
  - If/Else Statements



List Manipulation

Indexing:

```
>>> x = ['cs4440', 'is', 'cool']
>>> print(x[0])
cs4440

>>> x = ['cs4440', 'is', 'cool']
>>> print(x[-1])
cool
```

Inserting:

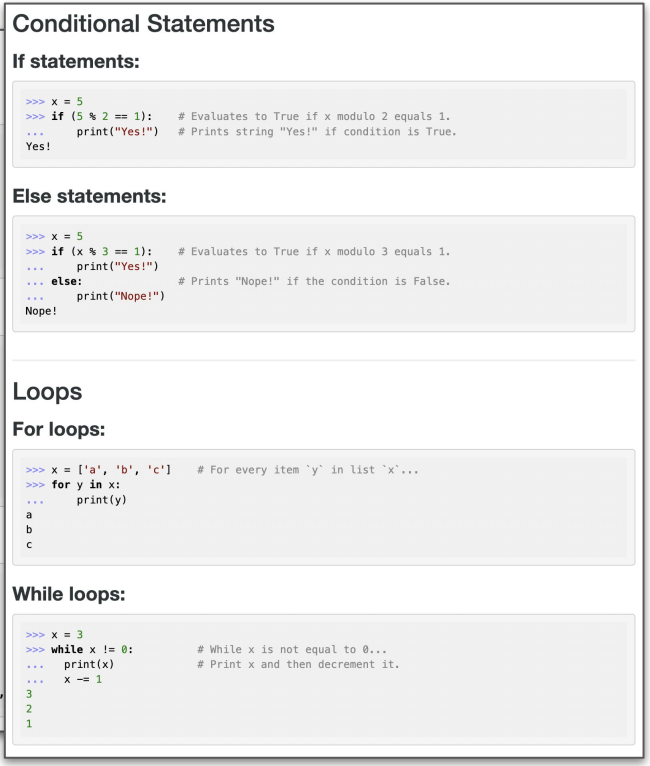
```
>>> x = ['cs4440', 'is', 'cool']
>>> x[-1] = 'fun'
>>> print(x)
['cs4440', 'is', 'fun']

>>> x.insert(2, 'super')
>>> print(x)
['cs4440', 'is', 'super', 'fun']
```

Joining:

```
>>> x = ['cs4440', 'is', 'cool']
>>> print(' '.join(x))
cs4440 is cool

>>> y = ['all', 'day']
>>> print(x + y)
['cs4440', 'is', 'super', 'cool',
```



Conditional Statements

If statements:

```
>>> x = 5
>>> if (5 % 2 == 1):    # Evaluates to True if x modulo 2 equals 1.
...     print("Yes!")   # Prints string "Yes!" if condition is True.
Yes!
```

Else statements:

```
>>> x = 5
>>> if (x % 3 == 1):    # Evaluates to True if x modulo 3 equals 1.
...     print("Yes!")
... else:               # Prints "Nope!" if the condition is False.
...     print("Nope!")
Nope!
```

Loops

For loops:

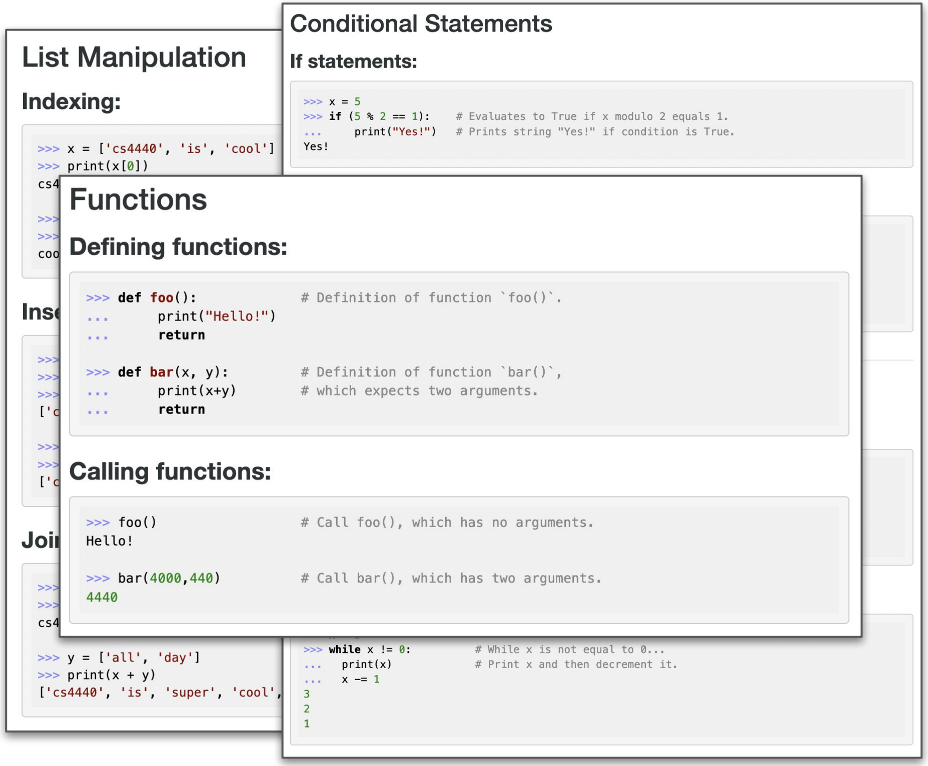
```
>>> x = ['a', 'b', 'c']    # For every item `y` in list `x`...
>>> for y in x:
...     print(y)
a
b
c
```

While loops:

```
>>> x = 3
>>> while x != 0:          # While x is not equal to 0...
...   print(x)             # Print x and then decrement it.
...   x -= 1
3
2
1
```

Other Key Concepts

- A few other concepts to review
  - Check these out in the **CS 4440 Wiki**
- Lists
  - Appending
  - Prepending
  - Insert, Remove
- Control Flow
  - Loops
  - If/Else Statements
- Functions



List Manipulation

Indexing:

```
>>> x = ['cs4440', 'is', 'cool']
>>> print(x[0])
cs4

>>>
>>>
coo
```

Inse

```
>>>
>>>
>>>
['c

>>>
>>>
['c
```

Joi

```
>>>
>>>
cs4

>>> y = ['all', 'day']
>>> print(x + y)
['cs4440', 'is', 'super', 'cool',
```

Conditional Statements

If statements:

```
>>> x = 5
>>> if (5 % 2 == 1):    # Evaluates to True if x modulo 2 equals 1.
...     print("Yes!")   # Prints string "Yes!" if condition is True.
Yes!
```

```
>>> while x != 0:           # While x is not equal to 0...
...   print(x)              # Print x and then decrement it.
...   x -= 1
3
2
1
```

Functions

Defining functions:

```
>>> def foo():                  # Definition of function `foo()`.
...       print("Hello!")
...       return

>>> def bar(x, y):              # Definition of function `bar()`,
...       print(x+y)            # which expects two arguments.
...       return
```

Calling functions:

```
>>> foo()                       # Call foo(), which has no arguments.
Hello!

>>> bar(4000,440)               # Call bar(), which has two arguments.
4440
```

Questions?

Debugging Your Code

Sample Program

- What will the following code do?

```
age = input("How old are you? ")
next_age = age + 1
print("Next year you will be", next_age)
```

What will the aforementioned code do?

Print a number (your age + 1) 0%

Print a string (your age + 1) 0%

All of the above! 0%

None of the above 0%

Sample Program

- What will the following code do?

```
age =
next_
print                    age)
```

Where to begin debugging?

- Errors say where the error is!
  - Filename
  - Line number
  - The actual line of code

```
Traceback (most recent call last):
  File "MyScript.py", line 2, in <module>
    next_age = age + 1
TypeError: must be str, not int
```

Where to begin debugging?

- Errors say where the error is!
  - Filename
  - Line number
  - The actual line of code

```
Traceback (most recent call last):
  File "MyScript.py", line 2, in <module>
    next_age = age + 1
TypeError: must be str, not int
```

Where to begin debugging?

- Errors say where the error is!
  - Filename
  - Line number
  - The actual line of code
- The error's root cause:
  - Program tried `"29"+1`
  - Strings and numbers are different data types!

```
Traceback (most recent call last):
  File "MyScript.py", line 2, in <module>
    next_age = age + 1
TypeError: must be str, not int
```

Where to begin debugging?

- Errors say where the error is!
  - Filename
  - Line number
  - The actual line of code
- The error's root cause:
  - Program tried `"29"+1`
  - Strings and numbers are different data types!
- **The fix:** cast age as an `int`

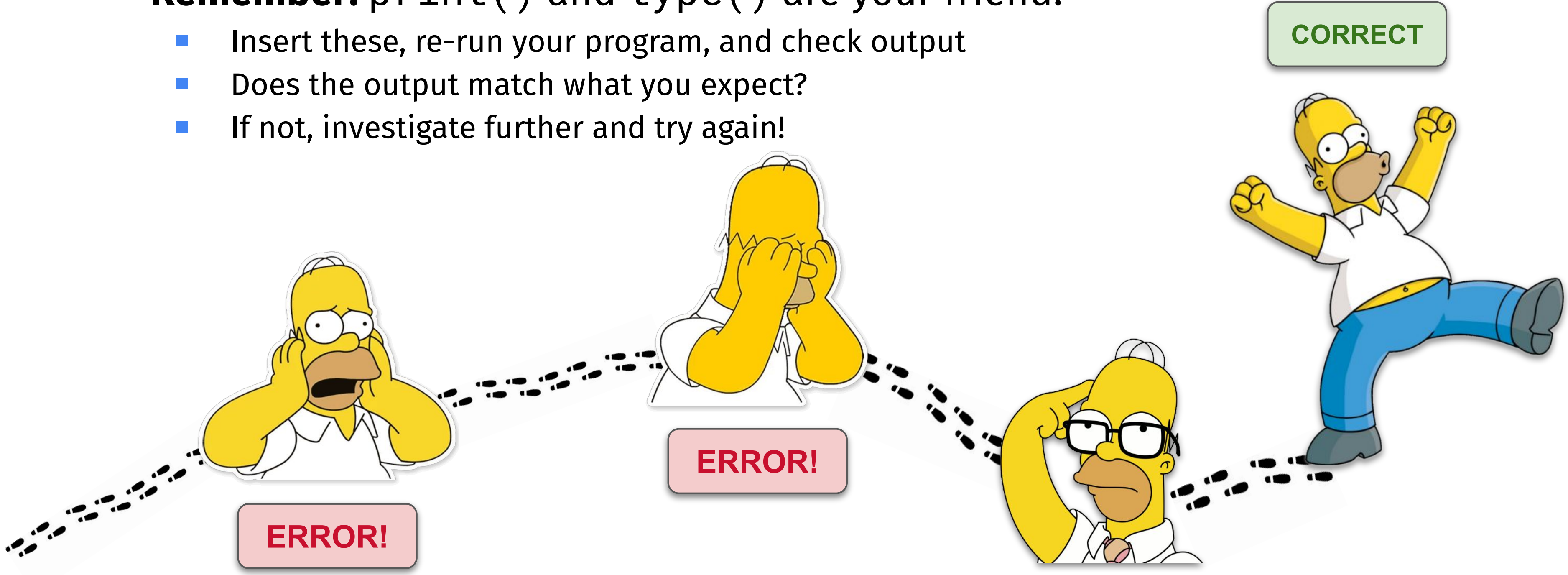
```python
age = input("How old are you? ")
next_age = int(age) + 1
```

Debugging is a Process

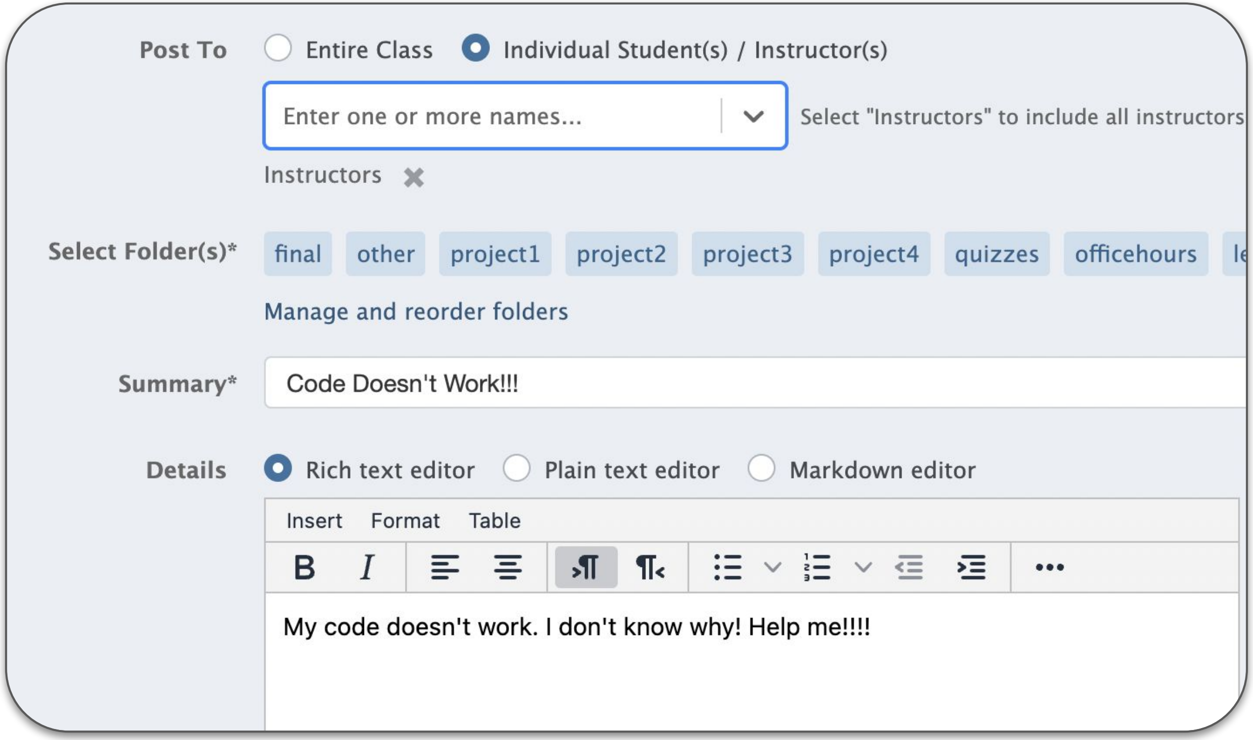
- **Remember:** `print()` and `type()` are your friend!
  - Insert these, re-run your program, and check output
  - Does the output match what you expect?
  - If not, investigate further and try again!

Debugging is a Process

- **Remember:** `print()` and `type()` are your friend!
  - Insert these, re-run your program, and check output
  - Does the output match what you expect?
  - If not, investigate further and try again!



Lazy Debugging



Post To ○ Entire Class [● Individual Student(s) / Instructor(s)]

Enter one or more names... Select "Instructors" to include all instructors

Instructors ✖

Select Folder(s)* final other project1 project2 project3 project4 quizzes officehours le

Manage and reorder folders

Summary* Code Doesn't Work!!!

Details [● Rich text editor] ○ Plain text editor ○ Markdown editor

My code doesn't work. I don't know why! Help me!!!!

Asking for Help

- **It’s perfectly fine to ask for help**
  - That’s what we / Piazza are here for!

Asking for Help

- **It's perfectly fine to ask for help**
  - That's what we / Piazza are here for!

- Help others help you! **Explain:**
  - What error code are you getting?
  - What do you think it means?
  - What fixes have you tried?
  - What fixes did not work?

Asking for Help

- **It's perfectly fine to ask for help**
  - That's what we / Piazza are here for!

- Help others help you! **Explain:**
  - What error code are you getting?
  - What do you think it means?
  - What fixes have you tried?
  - What fixes did not work?

- **Please** try to avoid **"instructor private posts"** about debugging your code
  - We get **a lot** of these near deadlines—it becomes impossible to keep up / help everyone!
  - We may un-private your post if it contains information that's useful for the class 🙂

Questions?

VM Setup

Virtual Machines (VM)

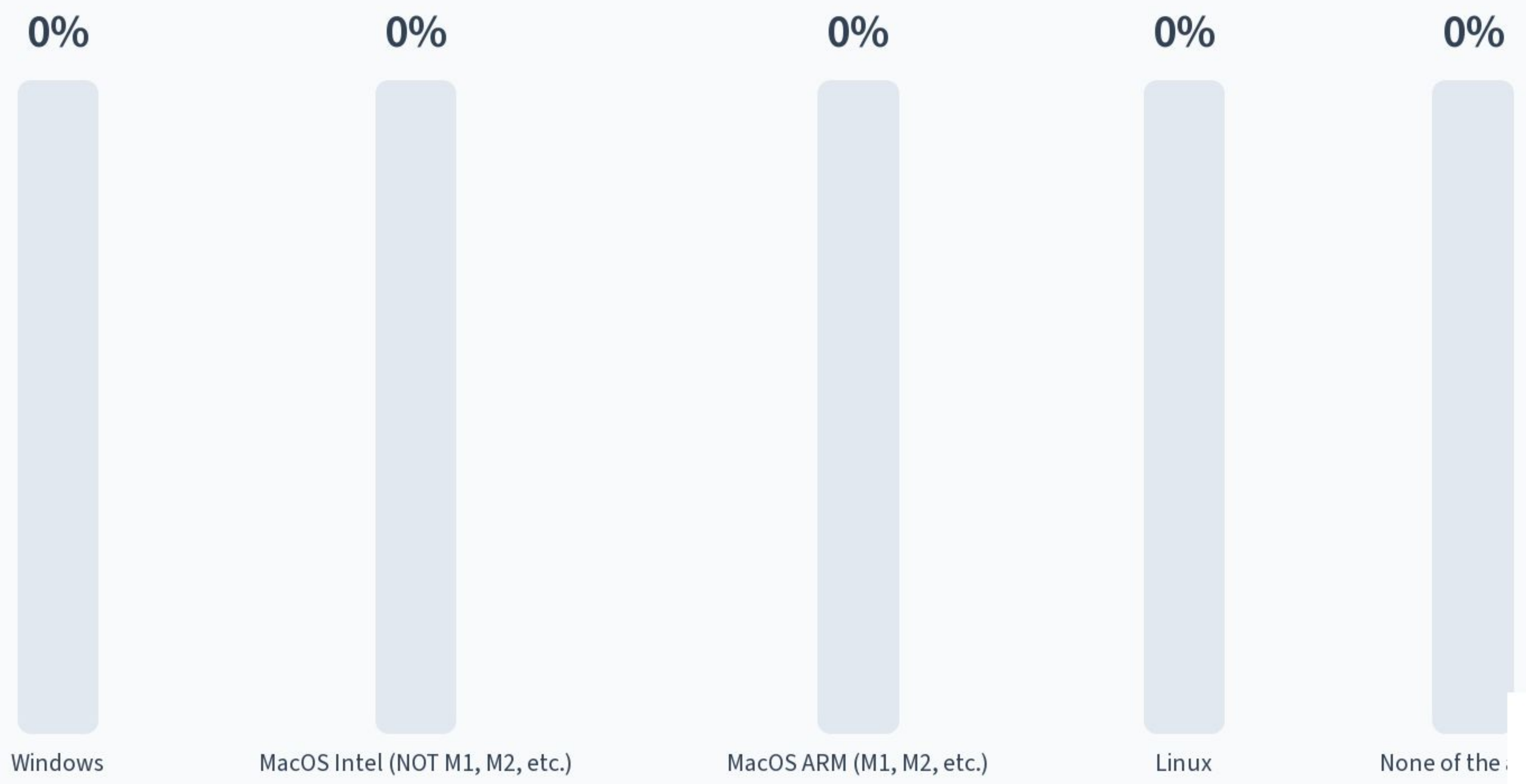
- Why do we use a **VM** in this course?
  - Minor software differences can **break your attacks**
  - We want everyone to have the **same software and OS**
    - Python & Firefox versions, security settings, etc.
  - We'll **grade** everyone using **this Linux VM environment**

Virtual Machines (VM)

- Why do we use a **VM** in this course?
  - Minor software differences can **break your attacks**
  - We want everyone to have the **same software and OS**
    - Python & Firefox versions, security settings, etc.
  - We'll **grade** everyone using **this Linux VM environment**

- **Important:** your computer determines what **VM software** you will use
  - Use **VirtualBox** if:
    - Your laptop is a **Windows-**, **Linux-**, or **Intel-based Mac** (i.e., **NOT** an M1/M2/etc.)
  - Use **UTM** if:
    - Your laptop is an **ARM-based Mac** (i.e., an M1/M2/etc.)

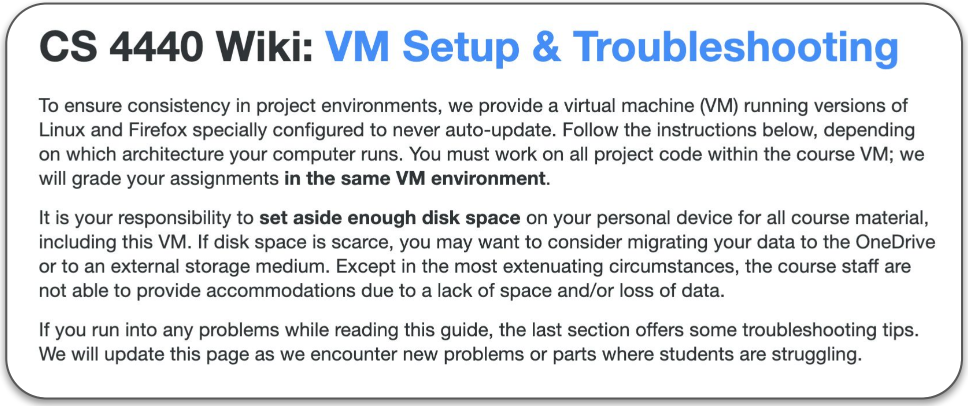
What kind of computer are you using?



Setup the CS 4440 VM

- Open the **CS 4440 Wiki**
  - See the **VM Setup** page
  - Follow the instructions
  - Once your VM is setup, you are free to leave!
  - In the meantime, feel free to **ask questions**

Course Homepage: **http://cs4440.eng.utah.edu**



CS 4440 Wiki: VM Setup & Troubleshooting

To ensure consistency in project environments, we provide a virtual machine (VM) running versions of Linux and Firefox specially configured to never auto-update. Follow the instructions below, depending on which architecture your computer runs. You must work on all project code within the course VM; we will grade your assignments **in the same VM environment**.

It is your responsibility to **set aside enough disk space** on your personal device for all course material, including this VM. If disk space is scarce, you may want to consider migrating your data to the OneDrive or to an external storage medium. Except in the most extenuating circumstances, the course staff are not able to provide accommodations due to a lack of space and/or loss of data.

If you run into any problems while reading this guide, the last section offers some troubleshooting tips. We will update this page as we encounter new problems or parts where students are struggling.

Next time on CS 4440...

Message integrity (a.k.a. applied cryptography)