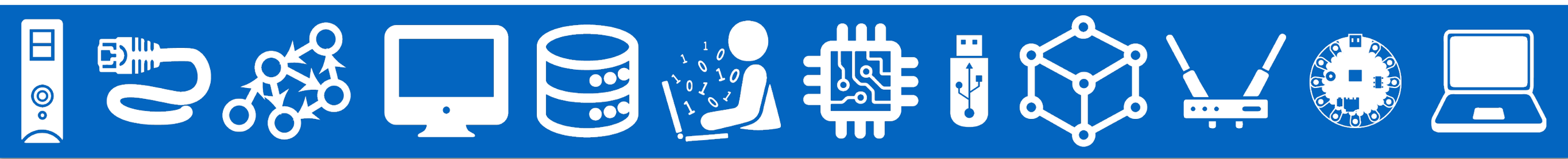
CS134 Lecture:
Wrapping Up



Announcements & Logistics

- **Lab 10** due Wed/Thurs at 10 pm
    - GREAT practice for the final exam!!
- CS134 Scheduled Final: **Wednesday, Dec 11, 9:30 AM**
    - Room: **Wachenheim B11 / Bronfman Auditorium**
- CS134 Review Session before Finals:
    - **Monday, December 9, 10a-12p**
    - Room: **TPL203**
- No TA Help Hours next week
    - Instructor Help Hours TBD

Do You Have Any Questions?

Announcements & Logistics

- **Final Exam: Dec 11, 9:30 am in Wachenheim B11**
  - **2 hour closed book exam**
  - Cumulative w/ more weight on topics post-midterm topics
  - Practice problems are posted; review lecture slides, lecture videos, POGILs, homework, and labs
  - Might consider reviewing 'Think Python' textbook on 'Resources' page on Course Website for specific reference questions
  - Exam Format will be very similar to midterm: open-ended + short answer mix

Do You Have Any Questions?

Last Time: Java vs. Python

- **Python** is a **loosely typed** language
  - *Why good?* Makes it easy to get started, less cumbersome / overhead
  - *Why bad?* Can lead to unexpected runtime errors, Python tries to "overcorrect" type issues whenever possible leading to unexpected behavior
- **Java** is a **strongly-typed** language: all variable types need to be declared at initialization and cannot change types
  - *Why good?* Can catch most type errors during compilation!
  - *Why bad?* Makes the code more verbose/requires more "boilerplate"

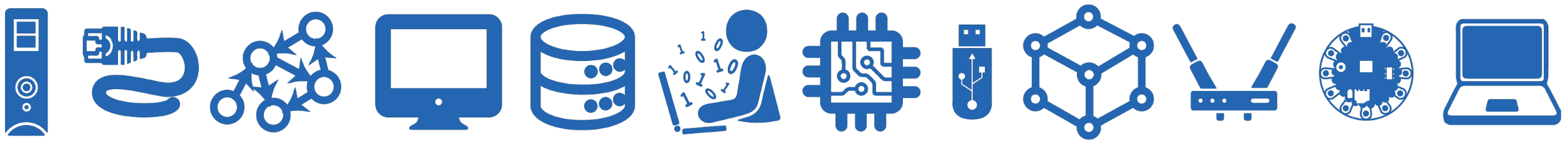
What we learned in this class isn't specific to Python, it applies to Java (and other languages) as well!

Today's Plan

- Learn about a cool **Python library** to do cool things with web data!
- Summarize **main topics** covered in CS 134 this semester
- How to do **more CS** stuff on your own/at Williams
- Complete **course evals**
  - We'll pause lecture for you to fill out course evals

Optional Fun Stuff:

Python & Webpages



What is a webpage?

"Ten movies streaming across that, that Internet, and what happens to your own personal Internet? I just the other day got... an Internet was sent by my staff at 10 o'clock in the morning on Friday. I got it yesterday [Tuesday]. Why? Because it got tangled up with all these things going on the Internet commercially. [...]

They want to deliver vast amounts of information over the Internet. And again, the Internet is not something that you just dump something on. It's not a big truck.
It's a series of tubes."

...not quite!

US Senator Ted Stevens (R-Alaska) in 2006, Head of the committee regulating Net Neutrality

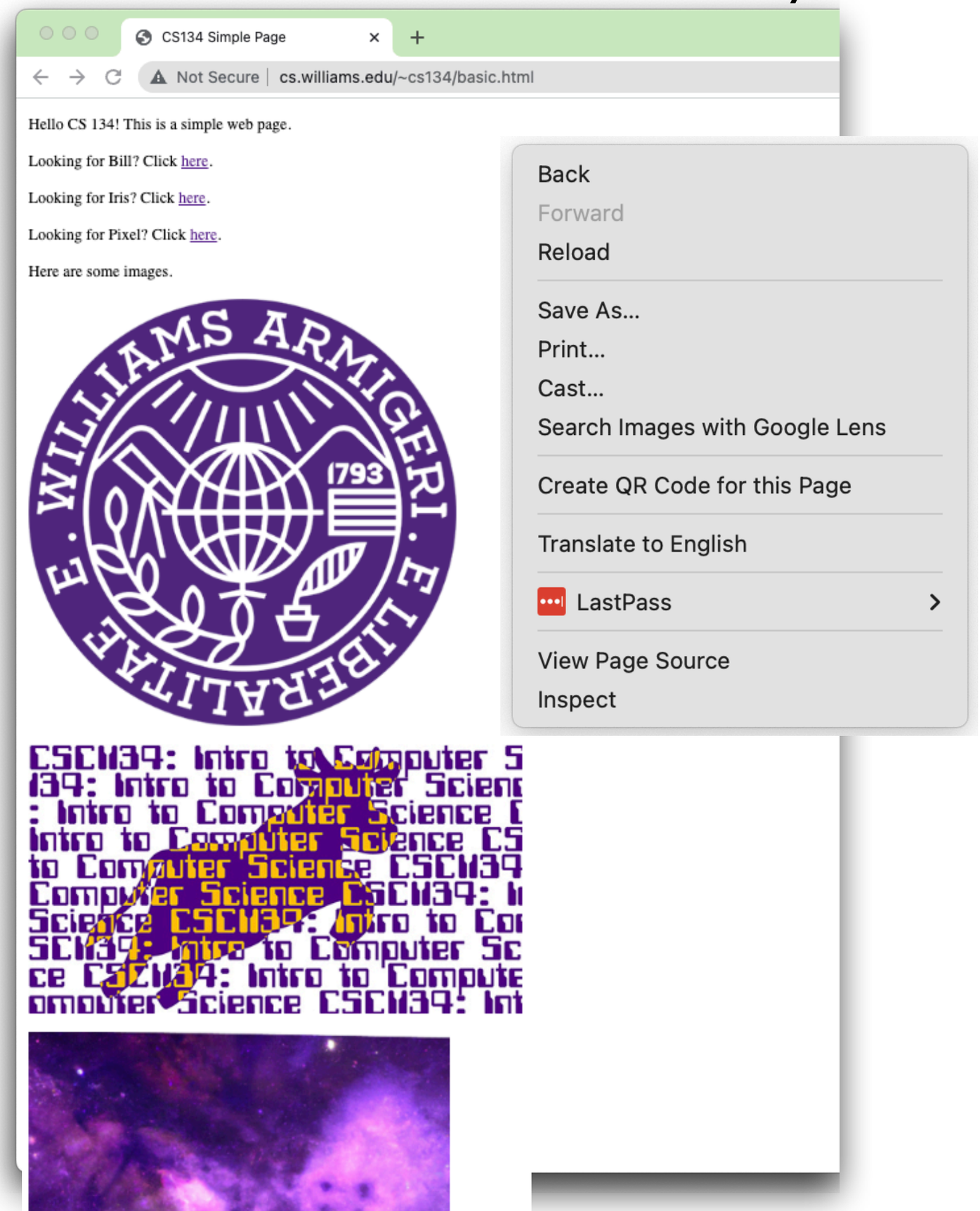
A webpage is just a publicly accessible file on a computer somewhere.

Learn more: https://youtu.be/AEaKrq3SpW8

HTML

- **H**yper**T**ext **M**arkup **L**anguage
- Specifies how to format text for your Internet Browser
  - Different tags/symbols specify how computer should display text
  - HTML is a markup language, not a programming language!

Try This...



- Right-click a webpage
- "View Page Source"

Try This...

```
<html>
  <head>
    <title>CS134 Simple Page</title>
  </head>

  <body>
    Hello CS 134!  This is a simple web page.

    <br><br>
    Looking for Bill?  Click <a href="http://www.cs.williams.edu/~jannen">here</a>.

    <br><br>
    Looking for Iris?  Click <a href="http://www.cs.williams.edu/~iris">here</a>.

    <br><br>
    Looking for Pixel?  Click <a href="https://www.cs.williams.edu/~iris/website/img/HAILab.jpg">
    here</a>.

    <br><br>
    Here are some images.
    <br><br>
    <img src="labs/images/img_wllmsLogo.png" alt="williams seal">
    <br><br>
    <img src="labs/images/img_courseLogo.jpg" alt="cs134">
    <br><br>
    <img src="labs/images/img_spaceCow.png" alt="space cow">

  </body>

</html>
```

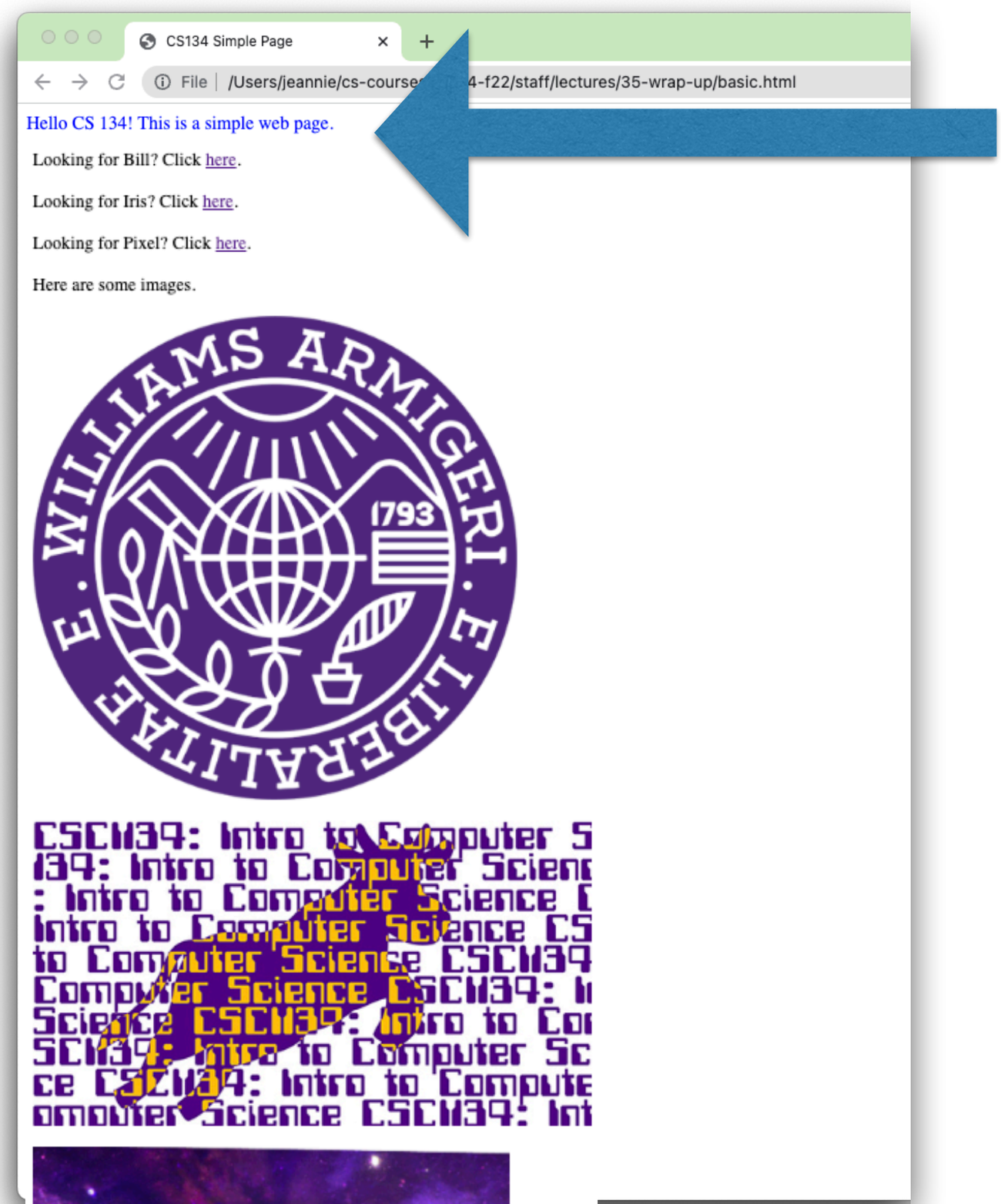
- Copy/Paste/Save with .html file extension in a text editor (like VS Code)

Try This...

```
<html>
  <head>
    <title>CS134 Simple Page</title>
  </head>

  <body>
    <font color="blue">Hello CS 134!  This is a simple web page.</font>

    <br><br>
    Looking for Bill?  Click <a href="http://www.cs.williams.edu/~jannen">here</a>.

    <br><br>
    Looking for Iris?  Click <a href="http://www.cs.williams.edu/~iris">here</a>.

    <br><br>
    Looking for Pixel?  Click <a href="https://www.cs.williams.edu/~iris/website/img/HAILab.jpg">
    here</a>.

    <br><br>
    Here are some images.
    <br><br>
    <img src="labs/images/img_wllmsLogo.png" alt="williams seal">
    <br><br>
    <img src="labs/images/img_courseLogo.jpg" alt="cs134">
    <br><br>
    <img src="labs/images/img_spaceCow.png" alt="space cow">

  </body>

</html>
```

- Make a small change. Save and view file in a web browser.

Try This...


CS134 Simple Page
File /Users/jeannie/cs-course 4-f22/staff/lectures/35-wrap-up/basic.html
Hello CS 134! This is a simple web page.
Looking for Bill? Click here.
Looking for Iris? Click here.
Looking for Pixel? Click here.
Here are some images.
WILLIAMS ARMIGERI
1793
LIBERALITAE
CSCI39: Intro to Computer S
Intro to Computer Scienc
Intro to Computer Science C
Intro to Computer Science CS
to Computer Science CSCI39
Computer Science CSCI39: I
Science CSCI39: Intro to Co
SCI39: Intro to Computer Sc
ce CSCI39: Intro to Compute
omputer Science CSCI39: Int

HTML

- <h1>Text goes here</h1> ➡ Makes a level1 heading
  - Guess: there's also an <h2></h2>, and <h3></h3>, and ...
- **Text goes here** ➡ Makes the text bold (also)
- *Text goes here* ➡ Makes the text italic (also)
- Link Text here ➡ Makes a hyperlink
- Text goes here ➡ Changes the font
  - Text goes here ➡ Changes font size
  - Text goes here ➡ Changes font color
- Text goes here ➡ Paragraph definition (~2 newlines)
-
 ➡ Line break (~1 newline)

HTML Header

- <html> ➡ Defines what markup language is being used
- <head> Text & Tags in here are part of the header </head>
- <title> This title appears in the web browser </title>
- <body> Text & Tags in here are part of the body text </body>
- </html> ➡ Ends HTML file

```
<html>
  <head>
    <title>CS134 Simple Page</title>
  </head>

  <body>
    Hello CS 134!  This is a simple web page.

    <br><br>
    Looking for Bill?  Click <a href="http://www.cs.williams.edu/~jannen">here</a>.

    <br><br>
    Looking for Iris?  Click <a href="http://www.cs.williams.edu/~iris">here</a>.
```

Pulling Source Code from Web Pages

```
terminal% pip install requests
```

```
>>> import requests
>>> r = requests.get('http://www.cs.williams.edu/~cs134/basic.html')
>>> r.text
```

```
'<html>\n  <head>\n    <title>CS134 Simple Page</title>\n  </
head>\n\n  <body>\n    Hello CS 134!  This is a simple web page.
\n\n    <br><br>\n    Looking for Bill?  Click <a href="http://
www.cs.williams.edu/~jannen">here</a>.\n\n    <br><br>\n
Looking for Iris?  Click <a href="http://www.cs.williams.edu/
~iris">here</a>.\n\n    <br><br>\n    Looking for Pixel?  Click <a
href="https://www.cs.williams.edu/~iris/website/img/
HAILab.jpg">here</a>.\n\n    <br><br>\n    Here are some images.
\n\n    <br><br>\n    <img src="labs/images/img_wllmsLogo.png"
alt="williams seal">\n\n    <br><br>\n    <img src="labs/images/
img_courseLogo.jpg" alt="cs134">\n\n    <br><br>\n    <img
src="labs/images/img_spaceCow.png" alt="space cow">\n\n  </
body>\n\n</html>\n    \n\n    \n'
```

http://docs.python-requests.org/en/master/user/quickstart/

Processing Source Code from Web Pages

- If you want to parse the HTML text from a string, the Beautiful Soup module is recommended:
    - https://beautiful-soup-4.readthedocs.io/en/latest/
- `terminal% pip install beautifulsoup4`

Processing Source Code from Web Pages

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(r.text, 'html.parser')
>>> print(soup.prettify())
```

```
<html>
 <head>
  <title>
   CS134 Simple Page
  </title>
 </head>
 <body>
  Hello CS 134!  This is a simple web page.
  <br/>
  <br/>
  Looking for Bill?  Click
  <a href="http://www.cs.williams.edu/~jannen">
   here
  </a>
  .
  <br/>
  <br/>
  Looking for Iris?  Click
  <a href="http://www.cs.williams.edu/~iris">
   here
  </a>
```

Processing Source Code from Web Pages

```
>>> soup.title
<title>CS134 Simple Page</title>
>>> soup.title.name
'title'
>>> soup.title.string
'CS134 Simple Page'
>>> soup.title.parent.name
'head'
>>> soup.img
<img alt="williams seal" src="labs/images/img_wllmsLogo.png"/>
```

Processing Source Code from Web Pages

```
>>> soup.a
<a href="http://www.cs.williams.edu/~jannen">here</
a>
>>> soup.find_all('a')
[<a href="http://www.cs.williams.edu/~jannen">here</a>,
 <a href="http://www.cs.williams.edu/~iris">here</a>,
 <a href="https://www.cs.williams.edu/~iris/website/img/
HAILab.jpg">here</a>]
```

Extracting All URLs

```
for link in soup.find_all('a'):
    print(link.get("href"))
```

```
http://www.cs.williams.edu/~jannen
http://www.cs.williams.edu/~iris
https://www.cs.williams.edu/~iris/website/img/HAILab.jpg
```

Extracting All Image alt text

```
for image in soup.find_all('img'):
    print(image.get("alt"))
```

```
williams seal
cs134
space cow
```

See beautifulsoup4 documentation

Beautiful Soup then parses the document using the best available parser. It will use an HTML parser unless you specifically tell it to use an XML parser. (See Parsing XML.)

Kinds of objects

Beautiful Soup transforms a complex HTML document into a complex tree of Python objects. But

Lots more beautifulsoup4 can do!
Learning the importance of documentation!
https://beautiful-soup-4.readthedocs.io/en/latest/

```
soup = BeautifulSoup('<b class="boldest">Extremely bold</b>')
tag = soup.b
type(tag)
# <class 'bs4.element.Tag'>
```

Tags have a lot of attributes and methods, and I'll cover most of them in Navigating the tree and Searching the tree. For now, the most important features of a tag are its name and attributes.

Name

Every tag has a name, accessible as `.name`:

```
tag.name
# u'b'
```

If you change a tag's name, the change will be reflected in any HTML markup generated by Beautiful Soup:

```
tag.name = "blockquote"
tag
# <blockquote class="boldest">Extremely bold</blockquote>
```

Attributes

What are we doing?!

- So now we can *scrape* HTML data from webpages...
- ...and parse the data so we can pull out meaningful text...

Why might we want to pull source code from the web?

- Maybe you're:
    - building a web crawler, documenting all the webpages on the Internet so their text can be searchable...
    - a sports recruiter and you need to pull wins/losses data from local amateur leagues...
    - a designer building software to make stock market transactions based on the weather...
    - a PR firm tracking in vivo mentions of particular products or brands
    - a humanitarian gathering evidence on organized crime groups
    - an AI researcher trying to generate new paint color names

What are we doing?!

- Python has **lots** more accessible modules that do other fun things:
  - Play music
  - Process images
  - Generate text
  - Statistical operations
  - Among others!

```
from music import *
# create a middle C half note
note = Note(C4, HN)
Play.midi(note) # and play it!
```

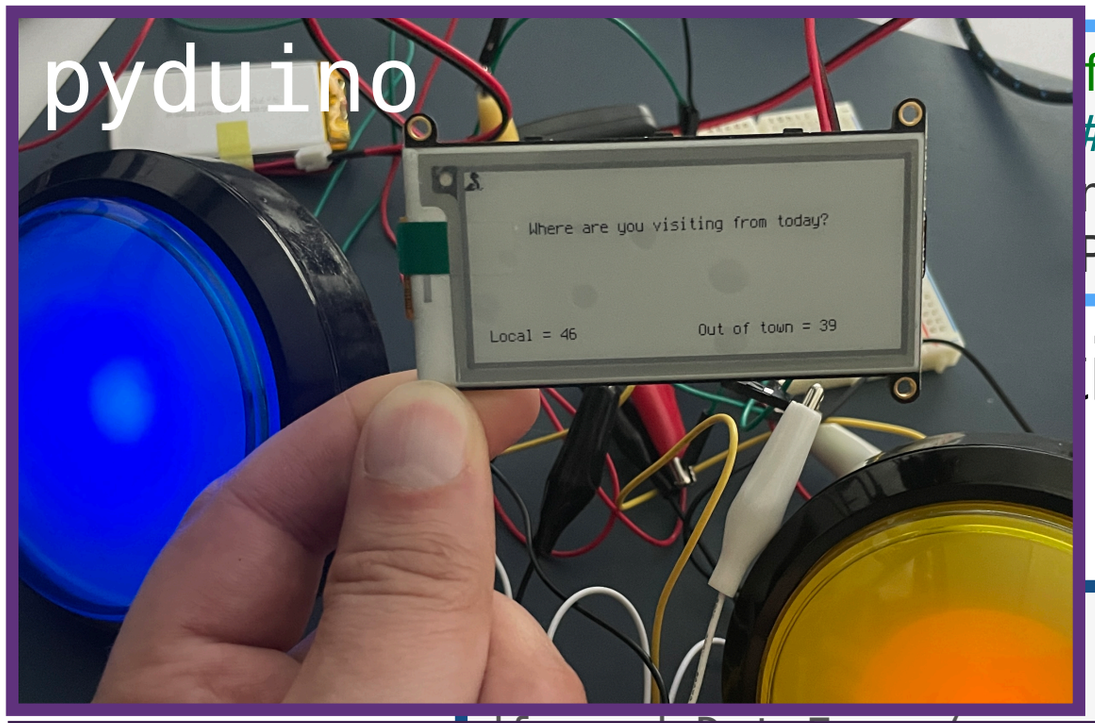
```
import matplotlib.pyplot as plt
from skimage import data,filters
image = data.coins()
# ... or any other NumPy array!
edges = filters.sobel(image)
plt.imshow(edges, cmap='gray')
```

```
import numpy as np
import pandas as pd
df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list("ABCD"))
s.value_counts()
```

```
from nltk import *
trigrams = list(ngrams(tokens, 3))
trigram_model = defaultdict(Counter)
for trigram in trigrams:
    trigram_model[(trigram[0], trigram[1])][trigram[2]] += 1
def generate_text(starting_words, model, num_words=20):
    sentence = list(starting_words)
    for _ in range(num_words):
        next_word = model[tuple(sentence[-2:])].most_common(1)[0][0]
        sentence.append(next_word)
    return ' '.join(sentence)
```

What are we doing?!

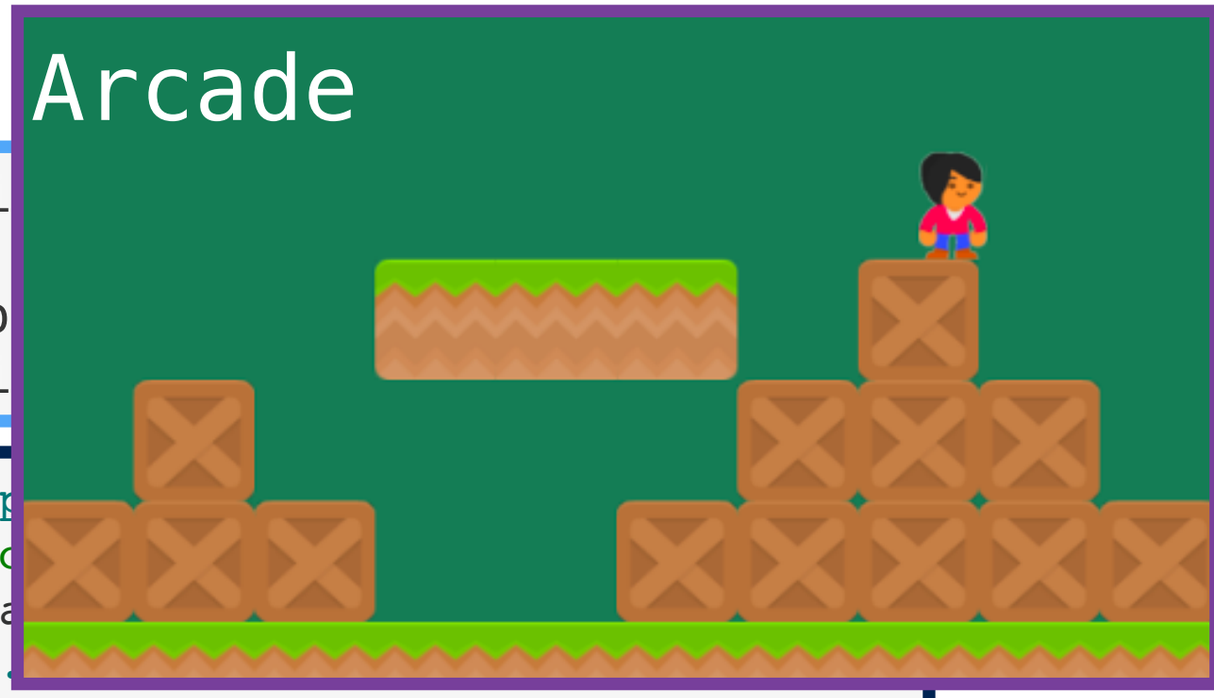
- Python has **lots** more accessible modules that do other fun things:

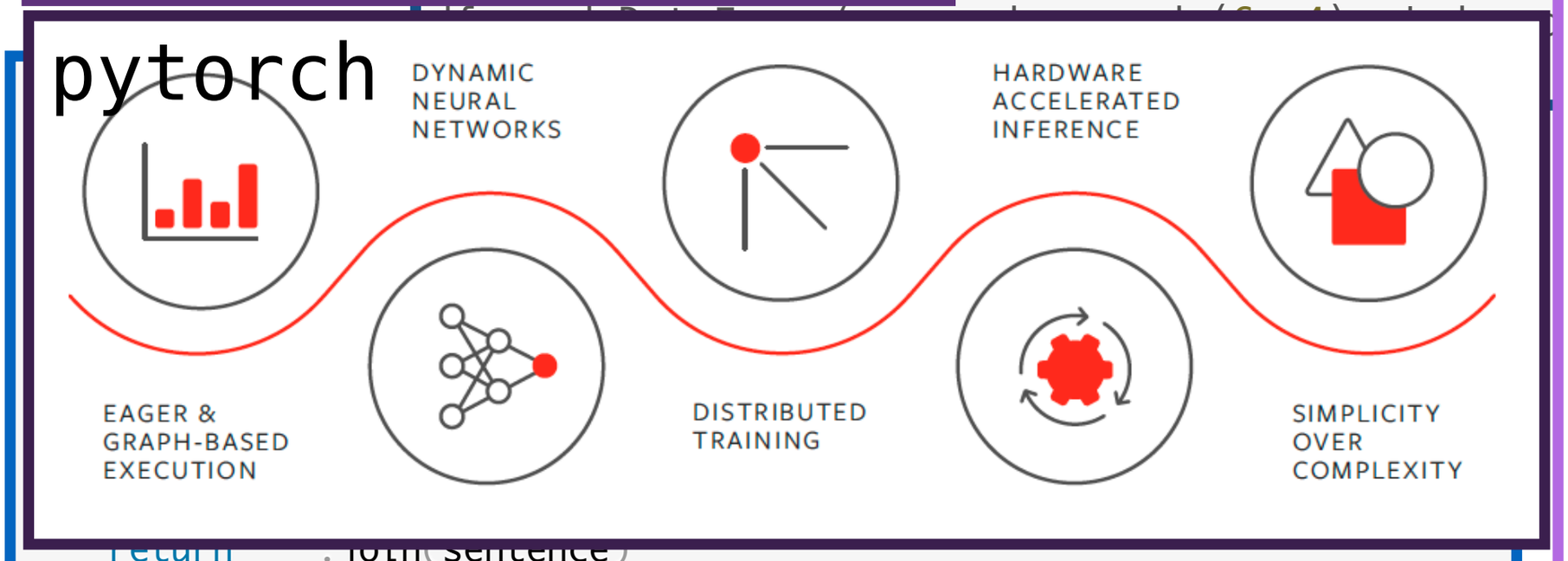


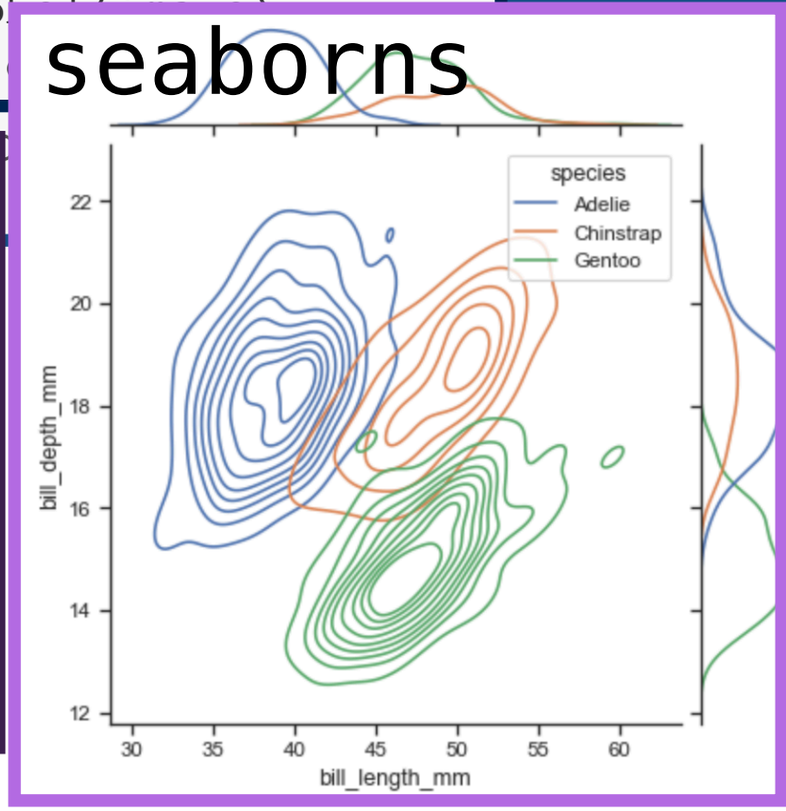

```
from musi
# create
note = No
Play.midi
```

ions

```
imp
fro
ima
# .
edges = filters.so
plt.imshow(edges,
```





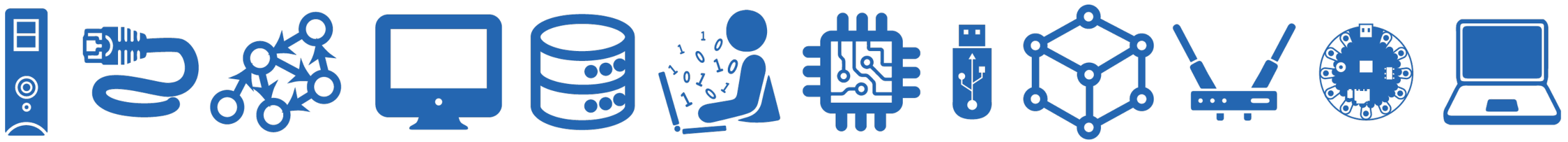


Take-away

- Python is a **powerful tool** that:
    - Processes, manipulates, organizes data
    - Accesses data
    - Creates beautiful things: art, solutions, puzzles, ...
    - Expands human capabilities

- But also: **communicates** complex computational ideas

Course Wrap-Up

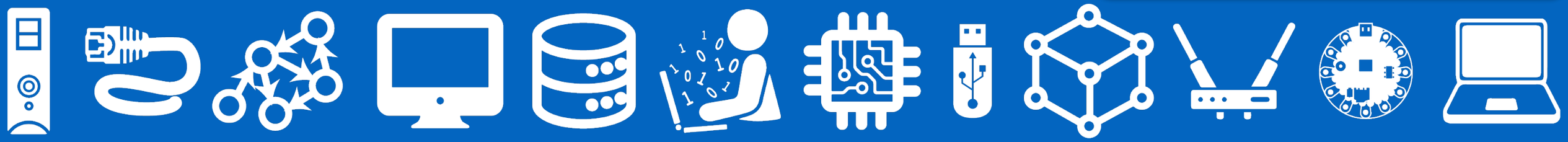


Remember when...

- We first learned **`input`**, casting to **`float`**, functions, and conditionals?!

```
def main():
    original_price = input("Enter the original cost of the item: ")
    sale_price = input("Enter the sale price: ")

    percent_reduced = percent_off(float(original_price), float(sale_price))

    print("Original price: $" + original_price)
    print("Sale price: $" + sale_price)
    print("Percent Off: " + str(percent_reduced) + "%")

    if percent_reduced >= 50:
        print("You got a great sale!")

def percent_off(orig, sa):
    return int((orig - sa)/orig * 100)

main()
```

Now we know how to combine these concepts with even more to solve complex problems!



CS134 in a Nutshell

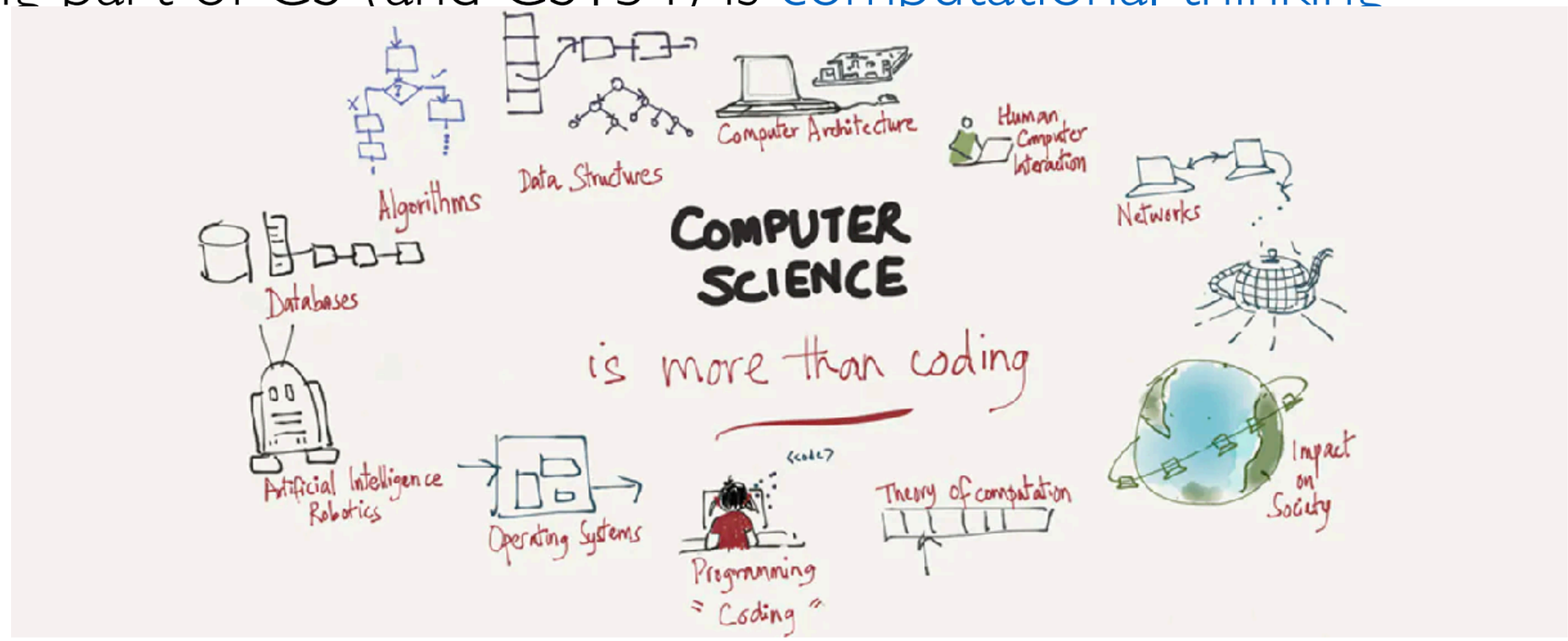
- We have covered many topics this semester!
- We started out learning the basics of programming, and we used python as our medium to explore these building blocks
- **Pre-midterm**
  - **Types & Operators** (int, float, %, //, /, concatenation, etc)
  - **Functions** (variable scope, return vs print, defining vs calling functions)
  - **Booleans and conditionals** (if elif else, >, <, ==, not, and, or)
  - **Iteration**: for loops, while loops, nested loops, accumulation variables in loops
  - **Sequences**: strings (operators, in/not in, iteration, etc) , lists (operators, indexing, slicing, etc), ranges, tuples, lists of lists
  - **Mutability** and **aliasing**
  - **Built-in python data structures**: lists, tuples and sets

CS134 in a Nutshell

- Then we moved on to more advanced CS topics
- **Post-midterm**
    - **New data structure**: dictionaries
    - **File reading**: with open(…) as, processing file lines in a loop
    - **Recursion**: recursive methods and classes
        - **Graphical recursion** with **turtle** graphics library
- **Classes, Objects, and OOP**
    - attributes, special methods, getters, setters, inheritance
    - "Bigger" OOP Examples: Autocomplete, Tic Tac Toe, Boggle, LinkedList
    - Special methods as well as sorted() with optional key argument
- **Advanced topics**:
    - Efficiency (Big-O), Linked Lists, Searching and sorting

Takeaway: What is Computer Science?

- Computer science ≠ computer programming!
- Computer science is the study of what computers [can] do; programming is the practice of making computers do useful things
- Programming is a big part of computer science, but **there is much more to CS** than just writing programs!
- A big part of CS (and CS134) is computational thinking



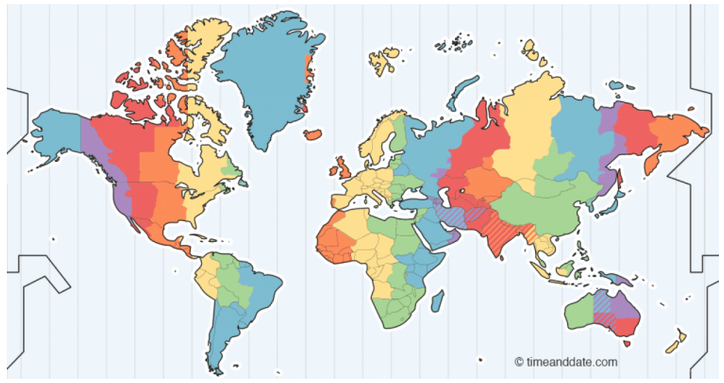
https://www.edsurge.com/news/2015-12-02-computer-science-goes-beyond-coding

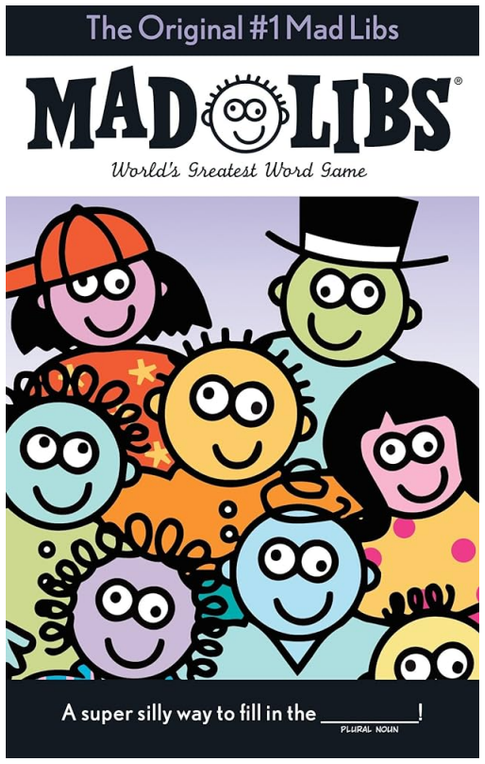
Biggest Takeaway: Computational Thinking

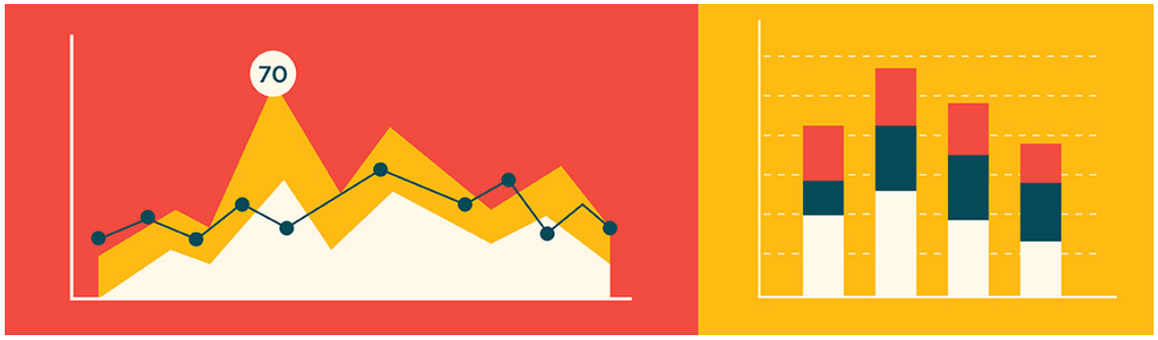
- Computational thinking allows us to develop solutions for complex problems. We present these solutions such that a computer, a human, or both, can understand.
- Four pillars of CT:
  - **Decomposition** - break down a complex problem into smaller parts
  - **Pattern recognition** – look for similarities among and within problems
  - **Abstraction** – focus on important information only, ignore irrelevant details
  - **Algorithms** - develop a step-by-step solution to the problem
- A computer can performs billion of operations per second, but computers only do exactly what you tell them to do!
- In this course we ~~will learn~~ **learned** how to 1) use CT to develop algorithms for solving problems, and 2) implement our algorithms through computer programs

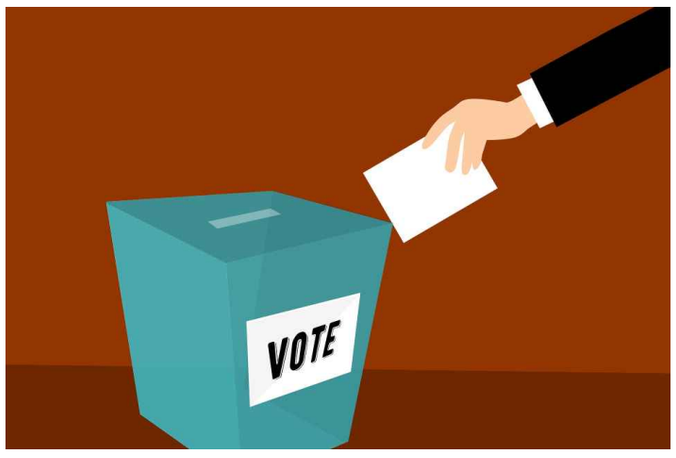
CS134 Labs: Practice with Computational Thinking

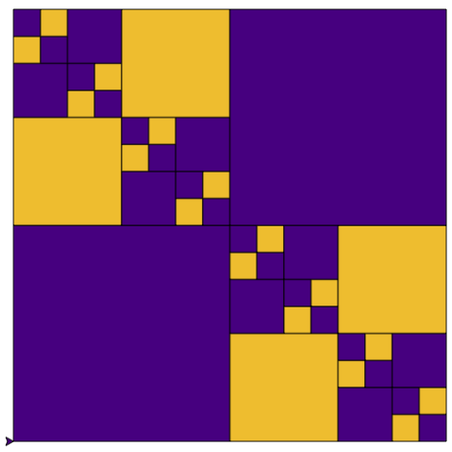
- Labs were designed to make look at real life **commonplace** processes through a computational lens







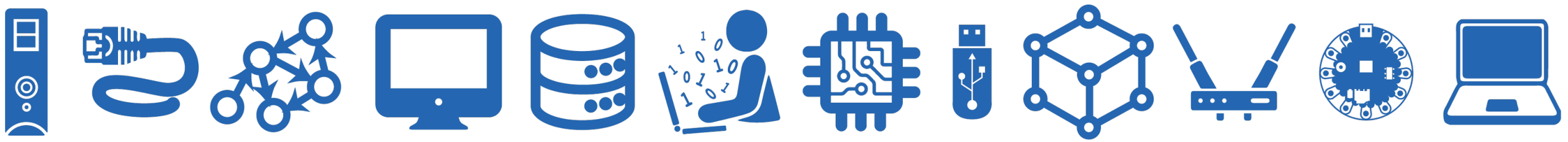




These Concepts Carry Over

- We used Python as a way to practice fundamentals of CS
    - Decomposition, Pattern recognition, Abstraction and Algorithms
- Programming languages just give us a way to express our logic
    - If the language changes, this expression changes (syntax)
        - But the outline of the solution (the logical steps) stay the same!
- Adapting to a new language is just a matter of getting familiar with its syntax, main structure and quirks
- Let's discuss this through high level comparison of Python vs Java

Beyond CS134



Beyond CS134

- For those interested in continuing on the CS path:
    - Obvious next step: take **CS136 + Math 200**
    - Practice more Java over winter break: redo our labs in Java!
- In general, if you enjoy **puzzles and programming**, there are many ways to practice these skills:
    - Try Project Euler: Math + CS puzzles
    - MIT course: The missing semester of your CS eduction
- Staying connected with CS as non-majors:
    - Can still take CS136 and other courses!
    - Winter Study: *Unix & Software Tools* and *Designing for People*
    - Come talk to us for more ideas!

What's Next?

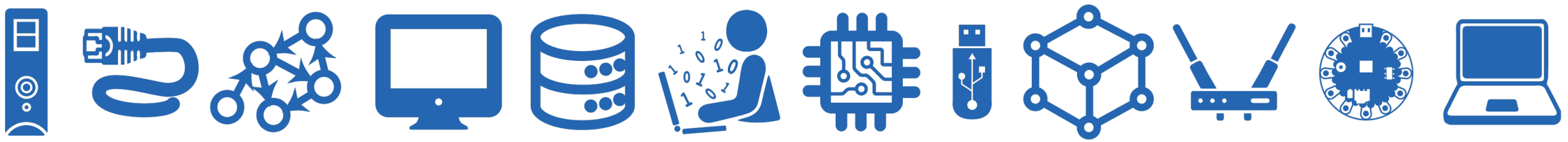
- If you liked coming up with your own algorithms and you **enjoyed the "puzzle"** aspects of labs, **CS 256** is for you!
  - *How to*: apply different algorithmic paradigms and prove that algorithms are correct and efficient
- If you're curious **how computers work**, how data is represented in memory, how software and hardware interface, **CS 237** is for you!
  - *How to*: optimize the practical parts of your program, get the most out of your physical computing resources, become a "hacker"
- If you enjoyed the process of learning python and want to better understand the **design choices of the language** itself, **CS 334** is for you!
  - *How to*: program in different language paradigms and pick the best language for the job (or design your own!)

Takeaways

- You all should be proud of how much you've learned!
- Computer Science is all about breaking down the problem and figuring out how to put the pieces together
    - This problem-solving mindset transcends languages/ majors, and will help you throughout your life!
- **Thank you** for your patience and enthusiasm throughout the course

WE MADE IT!

Student Course Surveys



Course Evals Logistics

- Two parts: **(1) SCS form**, **(2) Blue sheets** (both online)
- Your feedback helps us improve the course and shape the CS curriculum
    - Your responses are **confidential** and we only receive anonymized comments after we submit our grades
    - We appreciate your constructive feedback
- **SCS forms** are used for evaluation, **blue sheets are open-ended** comments directed only to your instructor

*To access the online evaluations, log into **Glow** (glow.williams.edu) using your regular Williams username and password (the same ones you use for your Williams email account). On your Glow dashboard you'll see a course called "**Course Evaluations**." Click on this and then follow the instructions you see on the screen. If you have trouble finding the evaluation, you can ask a neighbor for help or reach out to ir@williams.edu.*

The end!