CS134 Lecture 34: Wrap Up
Announcements & Logistics

- **Lab 10** due Wed/Thus at 10 pm
- **CS134 Scheduled Final:** *Friday, May 17, 9:30 AM*
  - Room: **TCL 123 (Wege)**
  - Reduced distraction room: **Bio 112**
- **CS134 Review Session** before Finals:
  - Wednesday May 15, **4.30-5.30 pm**
  - Room: **TCL 123 (Wege)**
- We will release Practice Final soon

Do You Have Any Questions?
Last Time: Sorting

• Discussed efficiency of selection and merge sort
  • You implemented and compared wall-clock time in Lab 10

• Takeaways:
  • Efficiency matters!
  • Big Oh is a good predictor of wall clock time
Today and Friday

• Today we will **wrap up the topics of CS134** (first 30 mins):
  • Overview of what we learned
  • Concepts vs programming language: discuss high level differences between Python vs Java
  • How to do more CS stuff beyond this class
• Last 15 or so mins: **course evaluations**
• Friday's plan:
  • **Jeopardy style review session of concepts!!**
    • Form teams of 5-6 students, come up w team names
    • Split up topics between teammates to maximize chance of winning
CS134 in a Nutshell

• We have covered many topics this semester!
• We started out learning the basics of Python and programming in general
• **Pre-midterm**
  • **Types & Operators** (int, float, %, //, /, arithmetic operators, etc)
  • **Functions** (variable scope, return vs print, defining vs calling functions)
  • **Conditionals** (if elif else and logical operators)
  • **Iteration**: for loops, while loops, nested loops, accumulation variables in loops
  • **Sequences**: strings, lists, ranges, lists of lists
  • **Mutability** and **aliasing**
  • **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 ... as, processing file lines in a loop
  
  • Recursion: recursive methods and classes
    
    • Graphical recursion with turtle graphics
  
  • Classes, Objects, and OOP
    
    • attributes, special methods, getters, setters, inheritance
    
    • “Bigger” OOP Examples: Autocomplete, Tic Tac Toe, Boggle
    
    • Special methods and associated operators/functions

• 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!

• Another part of CS is **computational thinking**

Take away: 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 perform billion of operations per second, but computers only do exactly what you tell them to do!

- In this course we will learn 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 look at real life commonplace processes through a computational lens.
Universal Skills and Toolkit

• Gained many skills that go beyond CS134/Python
• Navigating around your computer via Terminal
• Using git for collaboration
• Ability to utilize existing libraries for plotting/data visualization
• Practice on testing and debugging
• Skills that is useful irrespective of programming language
CS Concepts Carry Over

• We used Python as a tool to practice fundamentals of CS
  • Decomposition, Pattern recognition, Abstraction and Algorithms
• Programming language just gives us a way to express our logic
  • If the language changes, this expression changes
  • But the **algorithm (the logical steps)** stay the same!
Many Programming Languages

• Many programming languages out there
  • What is the most popular ones change over time
Adapting from One to Another

• Adapting to a new language is a matter of getting familiar with its syntax as well as practicing being "fluent" in it

• Let's discuss this through high level comparison of Python vs Java
# Python vs. Java

**Python**

- Powerful language used by many programmers
- Designed for making common programming tasks simple
- Good for new programmers, and for scientific computing

**Java**

- Powerful language used by many programmers
- Designed for building large-scale systems design
- Good fit for large, scalable reliable software projects
Python vs Java: Hello World

- Python has low overhead to get started
- Java has more overhead upfront
  - Needed to ensure declaring classes and types from get-go

```python
# hello.py
print("Hello, World!")
```

```java

# Hello.java

class Hello {
    public static void main(String args[]) {
        System.out.println("Hello, World!");
    }
}
```

we can call the function `print` without needing to define a class

Every Java program must define a `class`, and all code is inside a `class`. All functions in Java are `methods` and must be called using dot notation.
Python vs Java: Running Our Code

- **Python** is an interpreted language: *interpreter* runs through the code line by line and executes each line: this can also be done interactively!
- **Java** is a compiled language: code must be compiled first (converted to machine code) before it is executed.

# hello.py
```python
print("Hello, World!")
```

% python3 hello-simple.py
Hello, World!

# Hello.java
```
public class Hello {
    public static void main(String args[]) {
        System.out.println("Hello, World!");
    }
}
```

% javac Hello.java
% java Hello
Hello, World!

% python3
>>> print("Hello World!")
Hello World!
Python vs Java: Data Types

- Both Python and Java have data types (Ints, Floats, Booleans, Char/String etc)

- **Python** is flexible about its type:
  - Loosey goosey (technical term: *loosely typed*) language
  - Makes it easy to get started, less cumbersome / overhead
  - Can lead to unexpected runtime errors, tries to "overcorrect" type issues whenever possible leading to unexpected behavior

- **Java** is a *strongly-typed* language: variables types need to be declared at initialization and cannot be changed
  - Makes the code more verbose /more overhead
  - But will catch most of these errors during compilation!
Downside of Loose Types

• Python tries to fix "type mismatches" by doing bizarre things at times

• Does this look familiar?

```python
word1 = ['hello']
word2 = 'world'

word1 += word2  # calls.append secretly
print(word1)

['hello', 'w', 'o', 'r', 'l', 'd']
```
Beyond CS134

• For those interested in continuing on the CS path:
  • Take **CS 136** or **MATH 200**
  • If you want to practice Java over break: redo CS134 labs in Java

• In general, if you enjoy **puzzles and programming**, you can practice these skills on your own:
  • [Project Euler](http://projecteuler.net) (Math + CS puzzles)
  • [LeetCode](https://leetcode.com) (Coding Interview Prep, Python/Java examples)
  • MIT course: [The Missing Semester of Your CS Education](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-002-introduction-to-computer-science-and-programming-fall-2011/)

• CS courses as non-majors: can still take CS136, Math 200, winter study courses (Video games, Lida's winter study, etc)
Takeaways

• CS is all about breaking down a complex problem into smaller pieces and figuring out how to put the solution back together
  • This **problem-solving mindset** is a very useful skill to have!
• You all should be proud of how much you’ve learned!
• **Thank you** for your patience and enthusiasm throughout the course

**WE MADE IT!**
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.