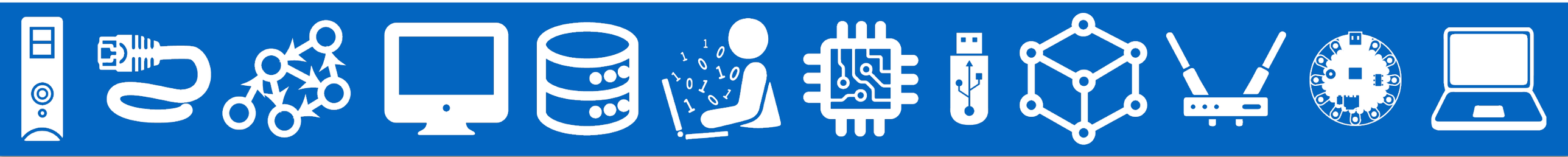
CS134 Lecture:
Python vs. Java



Announcements & Logistics

- **Lab 10** due Wed/Thus at 10 pm
- CS134 Scheduled Final: **Wednesday, December 11, 9:30 AM**
  - Room: **Wachenheim B11/Bronfman Auditorium**
    - All 3 sections will take the exam @ same time/place
- CS134 Review Session before Finals:
  - **Monday, December 9, time TBD**
    - Any constraints we should know about?

Do You Have Any Questions?

Last Time: Sorting Wrap Up

- Discussed "pythonic" approaches to common tasks
    - List comprehensions
    - tuple swapping
    - str.format()
- Nothing we talked about is a requirement
    - python-specific ways to approach particular problems

Today and Rest of Week

- Today we will discuss Java (as a stand-in for other non-Python languages)
- Wednesday we will wrap up the course
  - First 30 mins:
    - Overview of what we learned
    - Concepts vs programming language: discuss high level differences between Python vs Java, and why your CS134 skills will translate
    - How to do more CS stuff on your own/at Williams
  - Last 15 or so mins:
    - course evaluations
- Friday's class plan:
    - Jeopardy-style review session!!
    - Form teams with your classmates and come up with team names!
    - CS has a long tradition of bad puns and obscure references...

Why are there so many programming languages?

- Different languages are better at different tasks
  - Some languages hide the low-level details of the computer from the user to make the languages "easier to program" in
  - Some languages expose the low-level details of the computer to the user to make the languages "more powerful"
  - Some are designed to express algorithms in a particular way
    - functional, logic, object-oriented, statistical, etc.

There is no one language that is "the best"

Why Python in 134?

- CSCI 134 has been taught in other languages
  - C, Java, and Python most recently
- Python is a good language for *writing* code fast
  - There is very little "boilerplate", and the shortest Python programs can be written in just 1 line
  - Interactive Python is a great tool for quickly testing code snippets
  - OOP is optional but supported
- Python simplifies common data manipulation tasks with helpful libraries
  - Reading and writing files
  - Manipulating strings
- Very popular and well-supported in different communities

Why not use Python?

- Python not the best language for writing *fast* code
  - Other languages may give you more control over low-level details, which is necessary to optimize performance
- Python is not the easiest language to debug
  - Features that make Python easier to write also make certain errors easier to introduce
- We often want to build on or use existing code
  - You may want to work in a context where Python is not the language being used by others in that context

We need to be able to adapt to other languages and workflows!

Comparing Java and Python: Commonalities

- Both languages support similar building blocks
  - Loops and conditionals (if/else, for loops and while loops)
  - Built-in data types for numbers, booleans, strings, arrays/lists
  - Classes and OOP
  - Function frame model and scope
  - Recursion
  - ...

The ideas we learned in Python carry over to Java, we just need to learn how to express them using new syntax!

Comparing Java and Python: Differences

- Java is a statically typed language
  - In Java, each variable must specify a type which *cannot be changed*
  - In Python, types are not specified, and a variable's type can change

```
x = 3
x = "hello"
```

```
int x = 3;
String y = "hello";
```

Pros and Cons of Strict Typing

- **Python** is a "Loosey goosey" (technical term: **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"

Example: Python's Loose Types

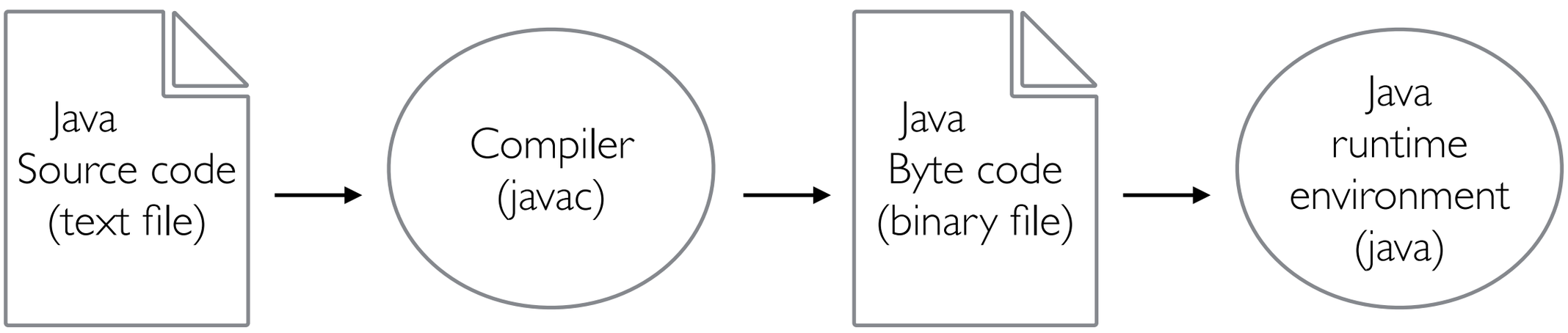
- Confusingly, Python tries to fix "type mismatches" by doing bizarre things
- Does this look familiar?

```
>>> word1 = ["hello"]
>>> word2 = "world"

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

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

This is not possible in Java. Java takes types very seriously...

Unlike Python, Java is ...

- Java is (in many senses) a compiled language
  - Java code you write is translated into bytecode
  - Bytecode is run in a Java virtual machine
    - There is no REPL (no equivalent of interactive python)
    - The Java virtual machine runs all the code in the "main method"



Compiling Can Be Helpful

- One consequence of the compiler is that certain type of errors can be found at *compile time*
  - This is almost like a round of debugging before there are even any bugs!

```
def add(x, y) :
 return x + y

...

add("abc", 3)
```

Manifestation of bug (run-time crash)

Location of bug

```
public int add(int x,
               int y) {
 return x + y
}
 ...

 add("abc", 3)
```

Compiler identifies type errors before program ever runs!

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
- **Interpreted** (line by line execution), allows for interactivity
- ***Dynamically typed***: Run-time error when variables are used incorrectly

Java

- Powerful language used by many programmers
- Designed for building large-scale systems design
- Good fit for large, scalable reliable software projects
- **Compiled**: must be ***compiled*** before execution, does not support interactivity
- ***Statically typed***: compile-time error when variables are used incorrectly

Neither language is "better" than the other. They are each useful for different things.

Python vs Java: Hello World

- Python has low overhead to get started
- Java has more overhead upfront (but we'll see why in CSCI 136)

```python
# hello.py

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

```java
# Hello.java

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

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

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

```
% python3 hello-simple.py
Hello, World!
```

```
% python3
>>> print("Hello World!")
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!
```

What's Next?

- If this is the last CS course you take, you can use Python to solve real problems!
  - A good way to practice is to use Python to accomplish interesting tasks (hobbies, course projects, ...)
- If you take CSCI 136, you will learn to write code that is reusable, maintainable, and scalable
  - More open-ended assignments that focus on design
  - Build your own data structures and learn to identify which data structure is the most appropriate for a given problem
  - Write code in Java!