Upcoming schedule
Tokenizer assignment
Compilers/interpreters

Zoom Wed
CoK Fri

One of the first steps in building an interpreter or a compiler is building a tokenizer
a token is a single unit of text in the language that is meaningful

---

Compilers/interpreters

A common modern approach involves multiple stages.

(Oracle) Java

$ javac Hello.java → Hello.class

→ <u>compiles</u> to a lower level language called "Java byte code"

$ java Hello (don't specify .class but it runs that file)

↳ <u>old</u> Oracle Java ran an interpreter on the bytecode to execute the program

---

python.org Python

$ python3 hello.py → hello.pyc (Python byte code)

compliles to

and then runs an interpreter on the pyc file

Why this complexity?

If you compile all the way to machine language, your compiled file won't work on diffrent processors.

So instead, compile to a processor-independent middle language, and then run an interpreter on it which has been designed for each processor

Another advantage is just separating diffrent jobs to diffrent programs

---

Back to Java: modern versions don't interpret the byte code, they do "just in time" compiling (JIT)

↗ compiles on-the-fly while program is running
