Welcome to CS450! (section 2)
High Level Languages
UMass Boston Computer Science
Instructor: Stephen Chang
Fall 2023
Welcome to CS450!
High Level Languages
UMass Boston Computer Science
Instructor: Stephen Chang
Fall 2023
What’s a Language?

• A language is for **communication**
  • With whom?

• A language is used to communicate to:
  • Other **people** (in a conversation)
  • To yourself (notes)
  • Across time!

This is a class about **language**

We will learn to use language to **communicate** (read, write, and speak) effectively
Welcome to CS450!

High Level Languages

UMass Boston Computer Science
Instructor: Stephen Chang
Fall 2023

What’s this?
What’s a Programming Language?

• A way for programmers to communicate ...
• ... machine instructions (to a computer)
  • i.e., “programs”
• ... ideas (to another programmer)
  • e.g., code review,
  • pull requests
• ... ideas (to themselves)
  • You are the most frequent reader of your code!
• ... across time!

Programs must be understandable by both computers and humans

In this class, we will learn to use language to code, read, write, and speak effectively
Welcome to CS450!

High Level Languages

UMass Boston Computer Science

Instructor: Stephen Chang

Fall 2023
CPU Language

Programmers don’t write machine code!

Because it’s difficult for humans to understand

Humans need “higher level” languages!

Every programming language is (primarily) for human communication
“high” level (easier for humans to understand)

English?

Q: Why don’t we just program in English?
A: It’s too imprecise

What about AI???

ChatGPT Getting Worse Over Time?

(ChhatGPT Getting Worse Over Time?)

Why Copilot writes bad code

Because of how language models work. They show how, on average, people write. They don’t have any sense of what’s correct or best code on GitHub is (by software standards) pretty old, and written by average programmers. Copilot spits out it’s best quality code there yet.

AI programming not quite there yet!

(Impossible! See cs 420)
“high” level (easier for humans to understand)

Q: Why don’t we just program in English?

A: It’s too imprecise

Code cannot be ambiguous

“low” level (runs on cpu)

Still needed in programs, for:
- Documentation
- Comments
- Specifications
(programs are more than code)

???

Machine code
“high” level
(easier for humans to understand)

This is easier for humans to understand, but what about the computer?

“low” level
(runs on cpu)

Language Level:

Assembly Language

More understandable feature:

Named instructions

Machine code
Less performant “high” level (easier for humans to understand)

This is easier for humans to understand, but what about the computer?

A higher-level language needs a **compiler** (another program!) to translate it to machine code

(Covered in another course!)

(usually)

**Assembler**

**Assembly Language**

Named instructions

More performant “low” level (runs on cpu)

**Machine code**

**Tradeoff:** This can introduce inefficiencies
Less performant  “high” level  
(easier for humans to understand)

(Covered in other courses!)

Programs are sequences of statements or “commands”  “imperative”

More performant  “low” level  (runs on cpu)

<table>
<thead>
<tr>
<th></th>
<th>“eval”</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>JavaScript, Python</td>
<td>C# / Java</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GC (no alloc, ptrs)</td>
<td>Classes, objects</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scoped vars, fns</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assembly Language</td>
<td>Named instructions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Machine code</td>
<td></td>
</tr>
</tbody>
</table>
Less performant  “high” level  
(easier for humans to understand)

“not imperative?”

More performant  “low” level  
(runs on cpu)

<table>
<thead>
<tr>
<th>Programs are sequences of statements or “commands”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“imperative”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>???</th>
<th>???</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript, Python</td>
<td>“eval”</td>
</tr>
<tr>
<td>C# / Java</td>
<td>GC (no alloc, ptrs)</td>
</tr>
<tr>
<td>C++</td>
<td>Classes, objects</td>
</tr>
<tr>
<td>C</td>
<td>Scoped vars, fns</td>
</tr>
<tr>
<td>Assembly Language</td>
<td>Named instructions</td>
</tr>
<tr>
<td>Machine code</td>
<td></td>
</tr>
</tbody>
</table>
Functional languages compute like this (combining arithmetic expressions) (instead of sequences of statements)

Function “call”

“variable”

“expression”

$f(x) = (x + 2)$

(but no “return”!)

$f(1) = ???

$f(2) = ???

$f(3) = ???

Is this programming?

Is this a programming language? YES!

This kind of programming is sometimes called “declarative”

“Declare” the computation you want. It’s “high level” because low-level details are omitted

(main topic in this course)
“high” level
(easier for humans to understand)

“declarative”

Describe computation with expressions
(compiler decides low level instructions)

“imperative”

Describe computation with exact sequence of statements

“low” level
(runs on cpu)

<table>
<thead>
<tr>
<th>Functional lang (Racket)</th>
<th>Expressions (no stmts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript, Python</td>
<td>“eval”</td>
</tr>
<tr>
<td>C# / Java</td>
<td>GC (no alloc, ptrs)</td>
</tr>
<tr>
<td>C++</td>
<td>Classes, objects</td>
</tr>
<tr>
<td>C</td>
<td>Scoped vars, fns</td>
</tr>
<tr>
<td>Assembly Language</td>
<td>Named instructions</td>
</tr>
<tr>
<td>Machine code</td>
<td></td>
</tr>
</tbody>
</table>
Lazy Arithmetic

$f(x, y) = x + 2$

$f(1, 2 + 3) = ???$

Lazy (functional) languages (also mathematical languages like R) delay computation until it’s needed (may cover in this course)

Result of this expression is not needed, so no need to compute it
“high” level (easier for humans to understand)

“declarative”

“imperative”

“low” level (runs on cpu)

<table>
<thead>
<tr>
<th>Lazy lang (Haskell, R)</th>
<th>Delayed computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional lang (Racket)</td>
<td>Expressions (no stmts)</td>
</tr>
<tr>
<td>JavaScript, Python</td>
<td>“eval”</td>
</tr>
<tr>
<td>C# / Java</td>
<td>GC (no alloc, ptrs)</td>
</tr>
<tr>
<td>C++</td>
<td>Classes, objects</td>
</tr>
<tr>
<td>C</td>
<td>Scoped vars, fns</td>
</tr>
<tr>
<td>Assembly Language</td>
<td>Named instructions</td>
</tr>
<tr>
<td>Machine code</td>
<td></td>
</tr>
</tbody>
</table>
Logic Programming – Even Higher Level

Why does this have to be the “input”?

\[ f(x) = x + 2 \]

\[ f(??) = 3 \]

\[ f(??) = 4 \]

“relational” programming

1 child_fact(oscar,karen,franz).
2 child_fact(mary,karen,franz).
3 child_fact(eva,anne,oscar).
4 child_fact(henry,anne,oscar).
5 child_fact(isolde,anne,oscar).
6 child_fact(clyde,mary,oscar).
7
8 child(X,Z,Y) :- child_fact(X,Y,Z).
9 child(X,Z,Y) :- child_fact(X,Z,Y).
10
11 descendant(X,Y) :- child(X,Y,Z).
12 descendant(X,Y) :- child(X,U,V), descendant(U,Y).
<table>
<thead>
<tr>
<th>Category</th>
<th>Example Languages</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification langs</td>
<td>Types? pre/post cond?</td>
<td></td>
</tr>
<tr>
<td>Markup (html, markdown)</td>
<td>tags</td>
<td></td>
</tr>
<tr>
<td>Database (SQL)</td>
<td>queries</td>
<td></td>
</tr>
<tr>
<td>Logic Program (Prolog)</td>
<td>relations</td>
<td></td>
</tr>
<tr>
<td>Lazy lang (Haskell, R)</td>
<td>Delayed computation</td>
<td></td>
</tr>
<tr>
<td>Functional lang (Racket)</td>
<td>Expressions (no stmts)</td>
<td></td>
</tr>
<tr>
<td>JavaScript, Python</td>
<td>“eval”</td>
<td></td>
</tr>
<tr>
<td>C# / Java</td>
<td>GC (no alloc, ptrs)</td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td>Classes, objects</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Scoped vars, fns</td>
<td></td>
</tr>
<tr>
<td>Assembly Language</td>
<td>Named instructions</td>
<td></td>
</tr>
<tr>
<td>Machine code</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Declarative languages can have imperative features, and vice versa

<table>
<thead>
<tr>
<th>Level</th>
<th>Example Languages</th>
<th>Key Features</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High</strong></td>
<td>Specification langs, Database (SQL), Logic Program (Prolog), Lazy lang (Haskell, R), Functional lang (Racket), JavaScript, Python</td>
<td>Types? pre/post cond?, tags, queries, relations, Delayed computation, Expressions (no stmts)</td>
</tr>
<tr>
<td><strong>Low</strong></td>
<td>C#, Java, C++, C, Assembly Language, Machine code</td>
<td>GC (no alloc, pointers), Classes, objects, Scoped vars, fns</td>
</tr>
</tbody>
</table>

### Goal
The goal is to learn "high-level" programming concepts, not a specific programming language.
Course Logistics

All course info available on web site:
https://www.cs.umb.edu/~stchang/cs450/f23
Racket  (main programming language for this course)

• Primarily “Functional”

• Easy (syntax) to learn
  • (But different than you might be used to!)

• Download at racket-lang.org/download
  • See hw0
  • Install and be ready to write code in next Monday’s 9/11 lecture

All course info available on web site:
https://www.cs.umb.edu/~stchang/cs450/f23
How to Design Programs, 2\textsuperscript{nd} ed.

Lessons:
\begin{itemize}
  \item Programs are also for \textit{high-level communication}
  \item This means that \textit{programs are more than just what the code does}
  \item Must be \textit{readable and explainable by others}
\end{itemize}

Available free at: \texttt{htdp.org}
\begin{itemize}
  \item Can buy paper copy (make sure it’s 2\textsuperscript{nd} ed) if you wish
\end{itemize}

All course info available on web site:
\url{https://www.cs.umb.edu/~stchang/cs450/f23}
Github

We will use github for code management

1. Create an account (free) if you don’t have one
2. Install a github client and learn basic commands
3. Tell course staff your account name
   • (see hw0)

All course info available on web site:
https://www.cs.umb.edu/~stchang/cs450/f23
HW 0

• 1\textsuperscript{st} part due: (this) Sunday 9/10 11:59pm EST
  • Create github account and learn basics
  • Tell course staff github account name (see hw0 details)
  • Install Racket
  • “Hello World”ish Racket programs
  • Be ready to program in class Monday

• 2\textsuperscript{nd} part due: Sunday 9/17 11:59pm EST
Other Infrastructure

• Gradescope
  • Submitting HW and grading

• Piazza
  • Non-lecture communication
Grading

• **HW: 80%**
  • Weekly: Out Monday, In Sunday
  • Approx. 12 assignments
  • Lowest grade dropped

• **Quizzes: 5%**
  • End of every lecture
  • To help everyone keep up

• **Participation: 15%**
  • Lecture, office hours, piazza

• No exams

• **A range: 90-100**
  • **B range: 80-90**
  • **C range: 70-80**
  • **D range: 60-70**
  • **F: < 60**

All course info available on web site:
https://www.cs.umb.edu/~stchang/cs450/f23
Grading

• **HW:** 80%
  • Weekly: Out Monday, In Sunday
  • Approx. 12 assignments
  • Lowest grade dropped

Evaluated on a program’s:

• **correctness**
  • i.e., test suites

• **readability**
  • Can someone read and explain what it does?

• **understanding**
  • Can you read and/or explain what it does?

All course info available on web site:
https://www.cs.umb.edu/~stchang/cs450/f23
Late HW

• Is bad ...
  • Grades get delayed
  • Can’t discuss solutions
  • You fall behind!

• Late Policy: 3 late days to use during the semester
HW Collaboration Policy

Allowed
- Discussing HW with classmates (but must cite)
- Using other resources, e.g., youtube, other books, etc.
- Writing up answers on your own, from scratch, in your own words / code

Not Allowed
- Submitting someone else’s answer
- It’s still someone else’s answer if:
  - variables are changed,
  - words are omitted,
  - or sentences rearranged …
- Using sites like Chegg, CourseHero, Bartleby, Study, ChatGPT etc.
Honesty Policy

• 1\textsuperscript{st} offense: zero on problem
• 2\textsuperscript{nd} offense: zero on hw, reported to school
• 3\textsuperscript{rd} offense+: F for course

\textbf{Regret policy}
• If you self-report an honesty violation, you’ll only receive a zero on the problem and we move on.
All Up to Date Course Info

Survey, Schedule, Office Hours, HWs, ...

See course website:

https://www.cs.umb.edu/~stchang/cs450/f23/
Check-In Quiz 9/6
(see gradescope)
(due right before midnight)