CMPT 140, 141, 143, 145: Introduction to Programming

6 Sep 2007
CMPT14x
Dr. Sean Ho
Trinity Western University

- *Pick up **syllabus***

TRINITY WESTERN UNIVERSITY

http://cmpt14x.seanho.com/

Outline for today

- Course information
  - Course website
  - Syllabus
  - Schedule
- Programming as problem-solving
  - Tools, toolsmiths, toolboxes
  - Top-down vs. bottom-up design
  - Example: woodcutting
- Demo of our Python programming environment

About CMPT 140, 141, 143, 145

- Everyone meets MWF 14:35-15:50
  - 140,145: also meet R 13:10-14:15, same room
- 141,143 run the whole semester
  - 140 runs the first six weeks only (to 25Oct)
  - 145 runs the last six weeks (but see assignments)
- Credit hours: 140=3, 141=4, 143=2, 145=2
- The usual sequence for most students is 140+145 (total of 5 credits), unless you're not planning to go further.

Course website

- http://cmpt14x.seanho.com/
- Also linked from myTWU/myCourses

- Note exam chs1-8 on W-Th 24-25Oct:
  - All attend (even those who don't regularly attend Thu section)
  - This serves as the final for CMPT140, midterm for 141/143

Lab sections

- Labs due every Wed via myCourses
- TAs will be available in the computing lab in Neufeld9
- Feel free to work in the lab at any open time
  - You have priority over other students when you're doing CMPT classwork
- Non-14x lab assistants are not prepped to answer your 14x questions
  - But they can handle printing problems, etc.
  - And most of them have taken 14x before

Outline for today

- Course information
  - Course website
  - Syllabus
  - Schedule
- Programming as problem-solving
  - Tools, toolsmiths, toolboxes
  - Top-down vs. bottom-up design
  - Example: woodcutting
- Demo of our Python programming environment

The Art of the Toolsmith

- Computers and software are tools; Computing scientists are toolsmiths
- The success of the tool is evaluated by the user, not by the toolmaker!

```
+   threadfn = create->threadfn;
+   data = create->data;
+
+   /* Block and flush all signals (in case we're not from keventd). */
+   sigfillset(&blocked);
+   sigprocmask(SIG_BLOCK, &blocked, NULL);
+   flush_signals(current);
+
+   /* By default we can run anywhere, unlike keventd. */
+   set_cpus_allowed(current, mask);
+
+   /* OK, tell user we're spawned, wait for stop or wakeup */
+   __set_current_state(TASK_INTERRUPTIBLE);
+   complete(&create->started);
+   schedule();
+
+   if (!kthread_should_stop())
+      ret = threadfn(data);
+
+   /* It might have exited on its own, w/o kthread_stop.  Check. */
+   if (kthread_should_stop()) {
+      kthread_stop_info.err = ret;
+      complete(&kthread_stop_info.done);
+   }
+   return 0;
```

≠

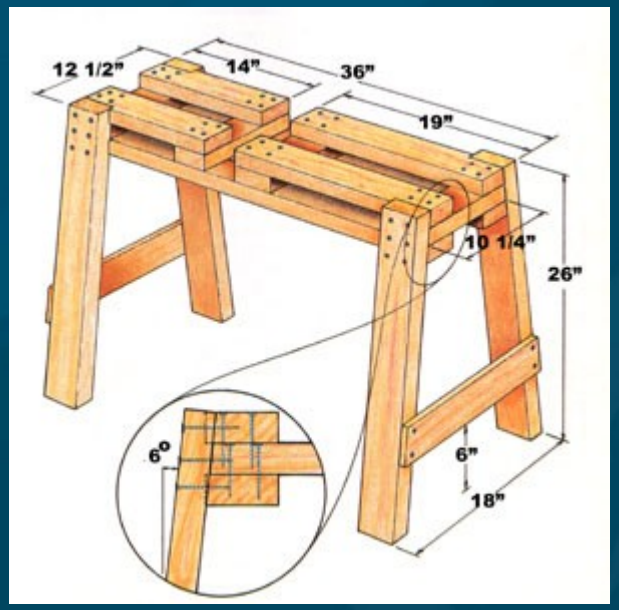
"the code is so beautiful!"

"does it do the job?"

Toolchains

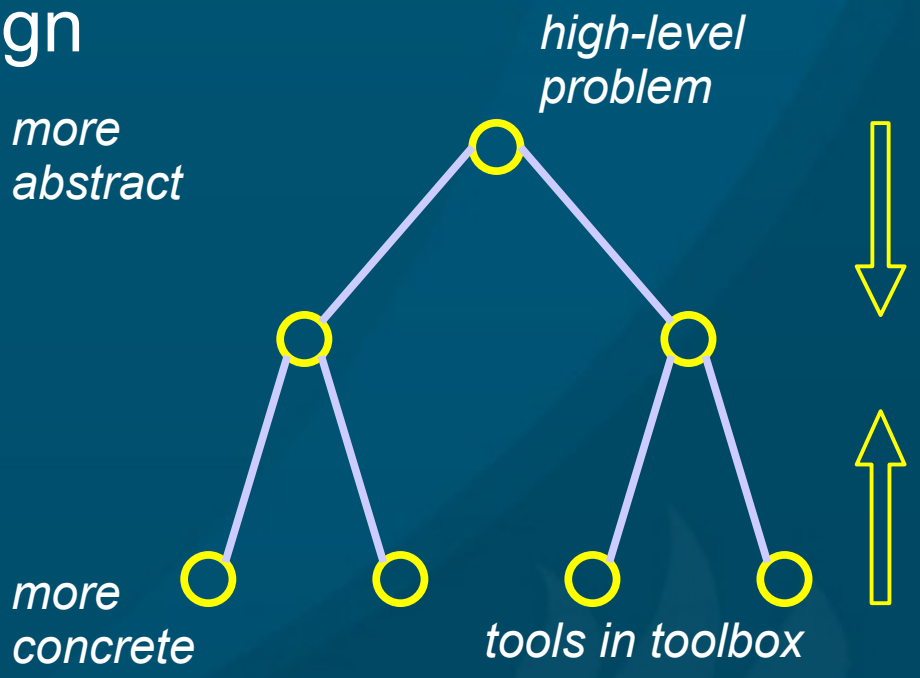
- Complex problems need sophisticated tools
- Complex tools are built up from simpler tools
- Always know what's in your toolbox:
  the tools you have to tackle problems
  - In software: libraries
  - In math: axioms
  - In philosophy: worldview, context

- In 14x: Python + libraries



Problem solving

- Top-down vs. bottom-up design



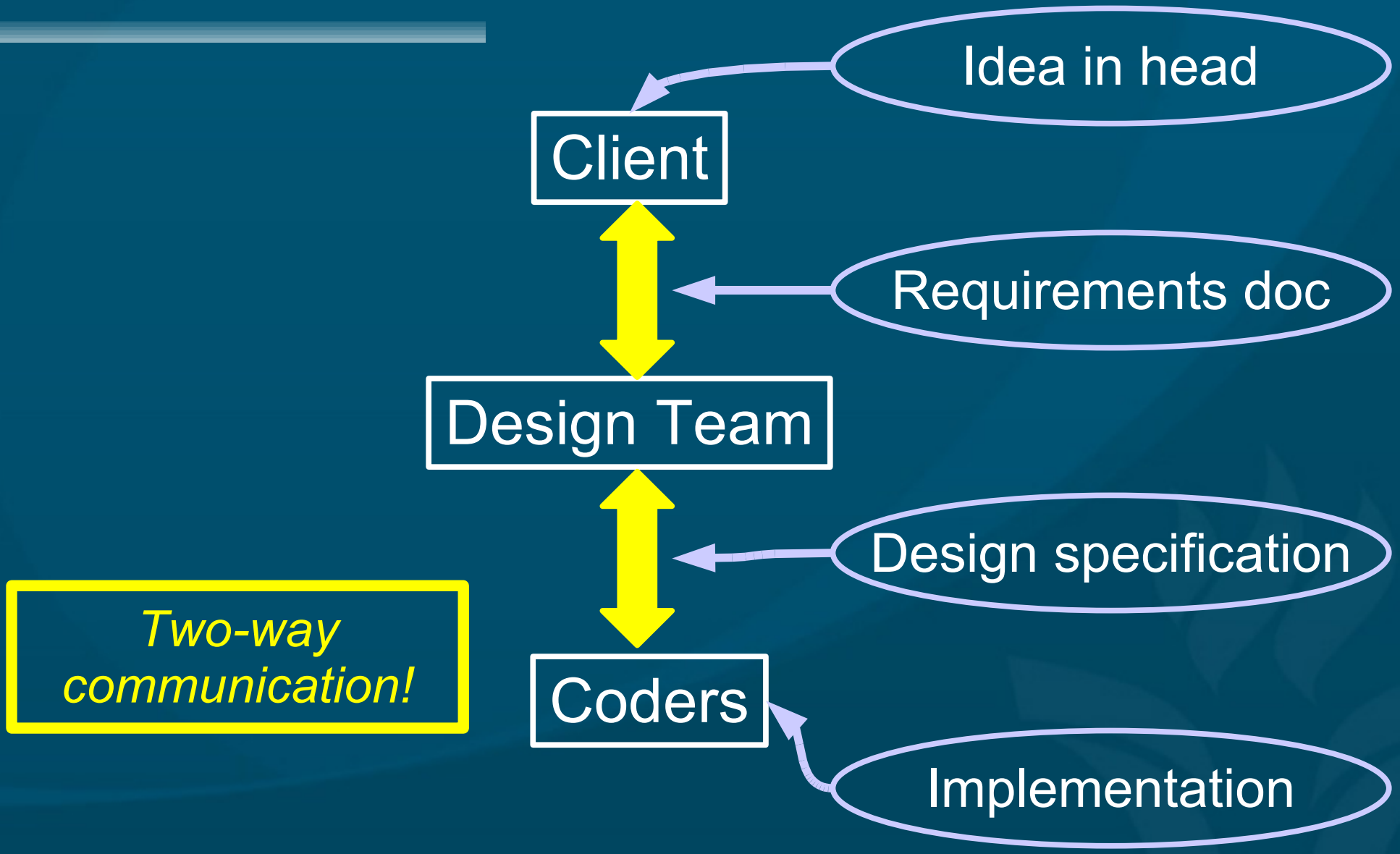
- ***W***rite everything down
- ***A***pprehend the problem
- ***D***esign a solution
- ***E***xecute the plan
- ***S***crutinize the results

Designing software vs. "hacking code"

- Look before you leap; think before you speak; ***design*** before you code!
- Programmer's optimistic schedule:
  - 4/5th coding
  - 1/5th testing/debugging
- Real-life schedule:
  - 1/3rd planning (***W***rite, ***A***pprehend, ***D***esign)
  - 1/6th coding (***E***xecute)
  - 1/2 testing/debugging (***S***crutinize)

Write
Apprehend
Design
Execute
Scrutinize

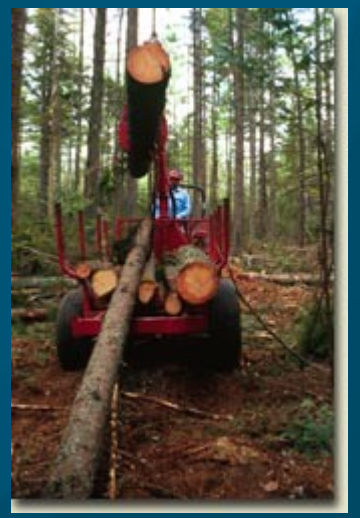
Interfaces in software development



Woodcutting example

- (see overheads / text pp.4-5)
- What are the library functions used in each version?

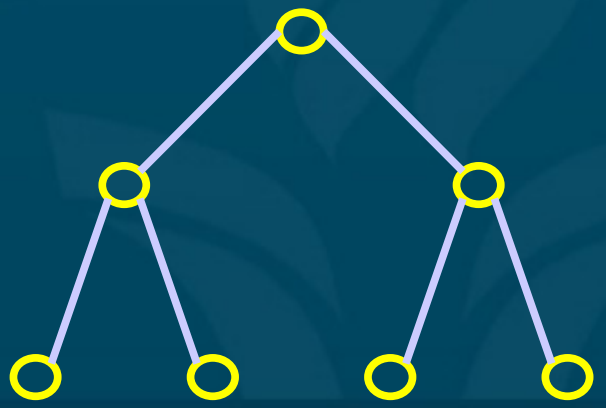
Woodcutting example



- We write out the solution in different levels of detail depending on
  - Who/what is executing the solution
  - What tools are available
- The solution is different for
  - An experienced lumberjack with good tools
  - A rookie who's never used a chainsaw
  - A software-controlled robot
  - A busy construction foreman
- *(which are more abstract / more concrete?)*

Review (1.1-1.4)

- Toolsmiths must know their toolboxes
  - (what does it mean for a computing scientist to be a toolsmith?)
- Top-down vs. bottom-up
- First step in problem-solving? (don't code yet!)
- ***WADES*** *(Write, Apprehend, Design, Execute, Scrutinize)*
- Levels of abstraction / levels of detail



Python/IDLE demo

- (demo of the Python programming environment)

Why Python?

- Why not M2, Java, C++, C#, PHP, Ruby, etc.?
- Syntax vs. semantics (more in a later section)
- At the CMPT14x level, the semantics of procedural programming in all these languages are pretty much the same
  - The only difference is syntax:

```
for (i=0; i<10; i++) {          (C++)
for i in range(10)              (Python)
```

- After this class, you'll be able to pick up any procedural language pretty quickly

TODO items

- Make sure you're registered for the right course:
  140+145, or 141, or 143
- Familiarize yourself with the course website:
  http://cmpt14x.seanho.com
- Do the Python/IDLE intro by next Wed
  (nothing to turn in, not graded)
- Read ch1 of the M2 text
- HW01 next Wed at start of class