

# Relationships in Software Development; Variables and Expressions

14 Sep 2010

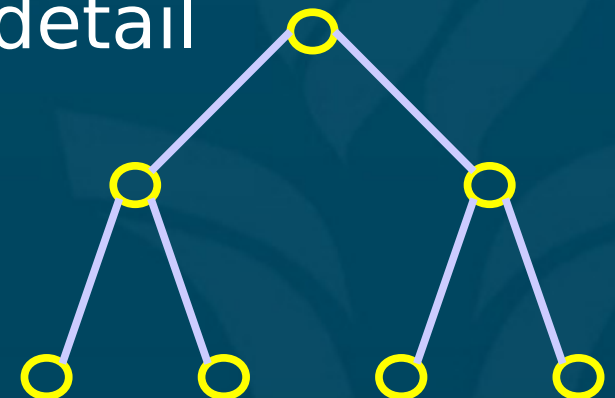
CMPT140

Dr. Sean Ho

Trinity Western University

# Review

- 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



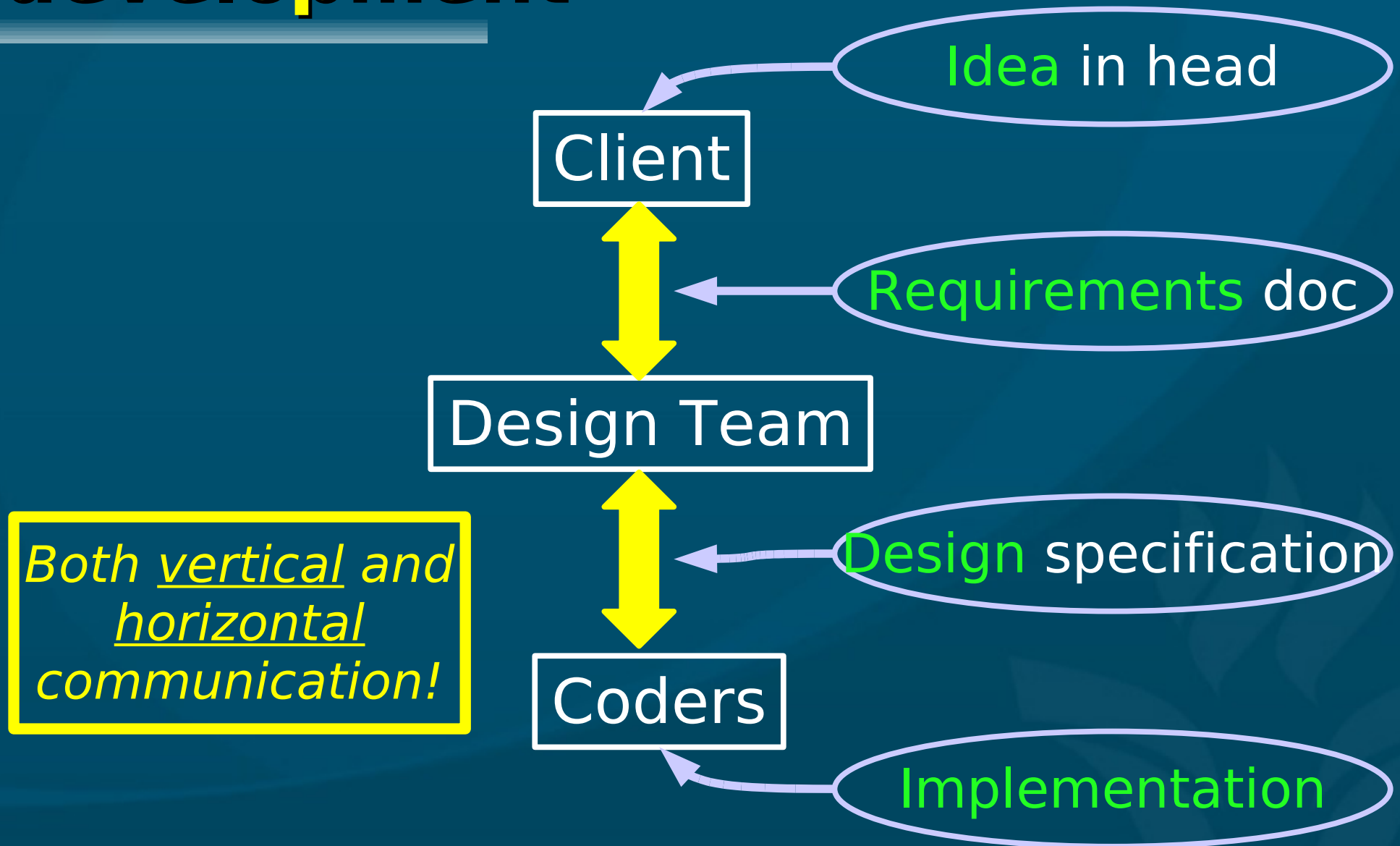
# Why Python?

- 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

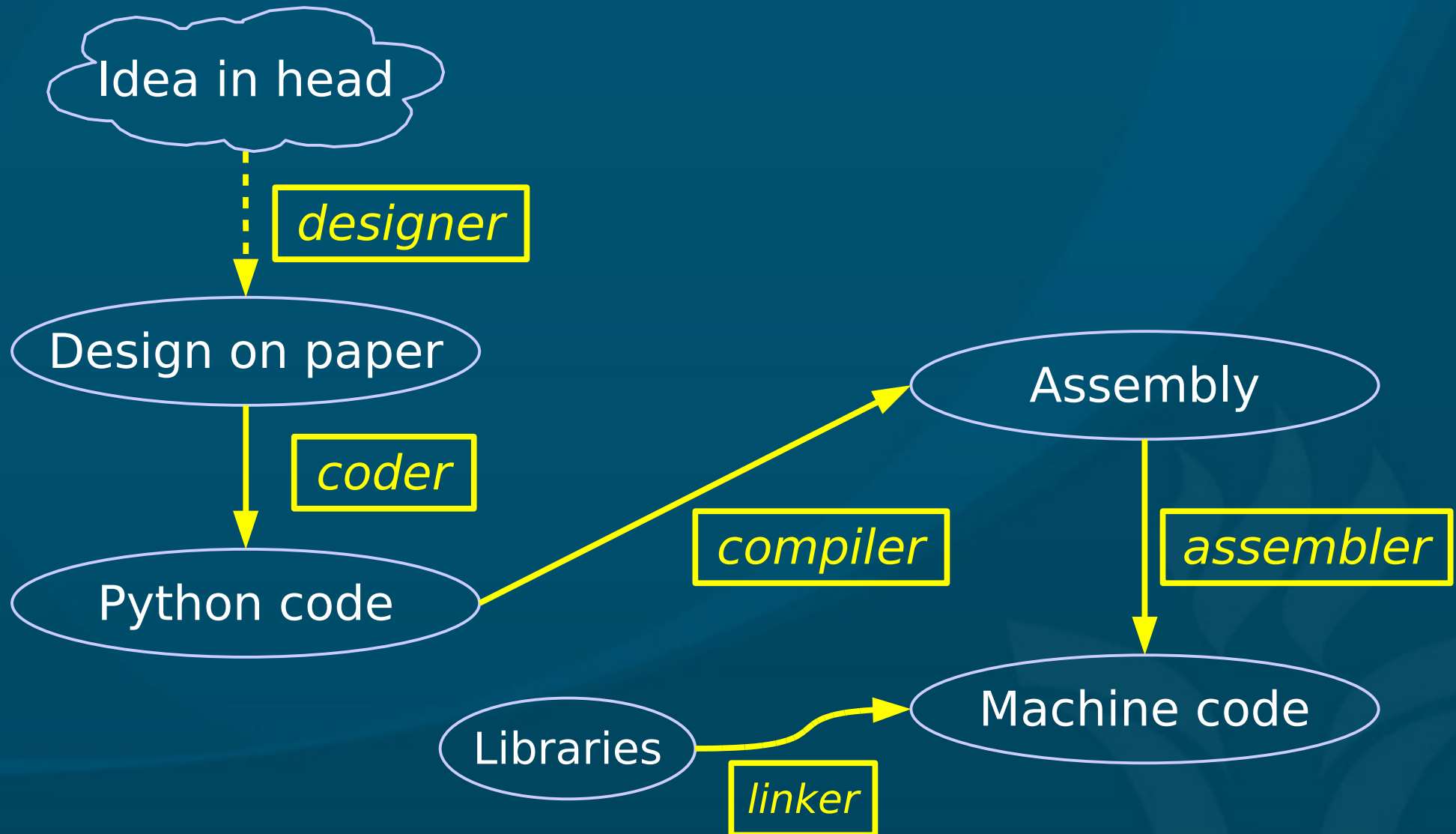
# Outline for today

- Roles and relationships in software devel.
- Abstractions: hardware, software, data types
- Variables and types
- Operators and expressions
  - Logical operators
  - Precedence
- Control abstractions: basics of a program

# Interfaces in software development



# Programming is translation



# “There's no 'I' in 'Team'!”

- Individual competency
  - Have something to **contribute**
  - Know how your niche **fits** in the whole
  - **Appreciate** other people's specialties
- Team competency
  - Mutual **trust** and respect
    - ◆ “Think of others as better than yourselves”
  - Self-**organization** into roles (may change)
  - **Initiative** – don't wait for others to do it
  - Constant **communication**

# Roles: producer vs. director

◆ *(This is just one way of organizing a team)*

## ■ Executive **Producer**

- Process, flow, keep team on-task, on-time

## ■ Technical **Director**

- Vision, artistic, technical integrity
- Prevent “feature creep”

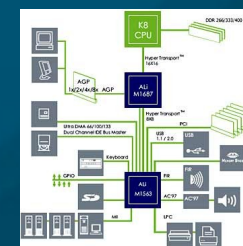
## ■ **Engineers**: implementers, make it happen

## ■ Architects/**designers**: give it purpose, make it beautiful

# Hardware abstractions

- Generally, most computers have these basic hardware components:

- Input
- Memory
- Processing
- Control
- Output



- Together with the software, the environment presented to the computer user by these is the **virtual machine**

# Software abstractions

- **Instructions:** basic commands to computer
  - e.g., ADD x and y and STORE the result in z
- **Programming language:** set of all available instructions
  - e.g., Python, C++, machine language
- **Program:** sequence of instructions
  - e.g., your “Hello World” program
- **Software:** package of one or more programs
  - e.g. Microsoft Word, Microsoft Office
- **Operating system:** software running the computer: provides environment for programmer
  - e.g., Windows XP, Mac OSX, Linux, etc.



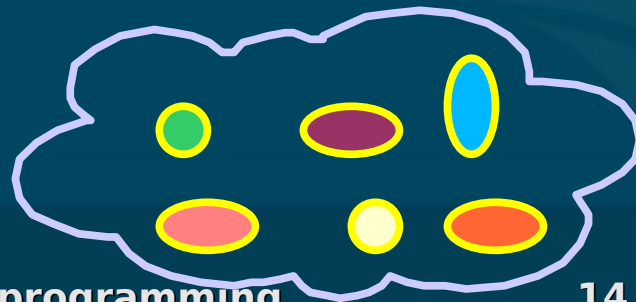
# Data representation

```
01000100100101110111010001001000
10100100110011110100100100001110
1011101110101010111110001001001
10111101011000001110001001111000
10001000100100101000111001000100
10010010100010101011101110101101
01010101001001010100010001001110
1111111110101010111110001000001
10101010000001011000010011111001
10000010001011110011010101000010
111110101010101010101010111001
00101000111000001111100010001011
```

- Data vs. **information**, knowledge vs. **wisdom**
- **Raw data** (factoids, memorized mantras) are useless unless you know what they **mean!**
- “There are 10 kinds of people in the world: those who know **binary**, and those who don't.”
  - (what does “**10**” mean?)

# Atomic vs. compound data

- **Atomic**: represents a single entity
  - e.g., 8,  $\pi$ ,  $6.022 \times 10^{23}$ , z
- **Compound**: entity that also is a collection of components: e.g.,
  - **Set**: {43, 5, -29.3}
  - **Ordered tuple**: (3,9) *(vs. set?)*
  - **Complex number**:  $4.63 + 2i$  *(set or tuple?)*
  - **Aggregate**: (name, age, address, phone#)
- **Singleton**: {43}



# Data types

- Certainly atomic vs. compound data are different types
- But even for atomic data there are **types**: e.g.
  - **Cardinals** (unsigned whole numbers; naturals): 0, 1, 2, 3, 4, 5, ...
  - **Integers** (signed whole): -27, 0, 5, 247
  - **Reals / Floats**: 5.0, -23.0,  $3 \times 10^8$
  - **Booleans**: True, False
  - **Characters**: 'a', 'H', '5', '='
  - **Strings**: "Hello World!", "5"

# Types in Python

- Python has many **built-in** types; here are some:
  - **int**: e.g., 2, -5, 0
  - **float**: e.g., 2.3, -42e6, 0.
  - **str**: e.g., 'hello', "world", '!', "
  - **bool**: True, False
  - **tuple**: e.g., (2, -1, 'hi'), ()
- You can find the **type** of an expression with:
  - `type(2.3)`
- A complete list of types is at <http://docs.python.org/ref/types.html>

# Review of today

- Relationships: requirements, design, implement
  - Roles: producer, director
- Abstractions: 5 HW, software terms, ADT
  - Contrast math “Reals” with Python float?
- Data types: atomic, compound (examples?)

# TODO items

- Familiarize yourself with the course website:  
<http://cmpt140.seanho.com>
- Do the **Python/IDLE** intro by Thu  
(nothing to turn in, not graded)
  - Lab1 is due the following Thu after that
- Read **ch1-2** of the textbook
- **HW02** due next Tues before start of class
  - Electronic turn-in: upload to myCourses