§1.6.5-§2.1: Software Abstractions and Control Structures

11 Sep 2006
CMPT14x
Dr. Sean Ho
Trinity Western University

devo

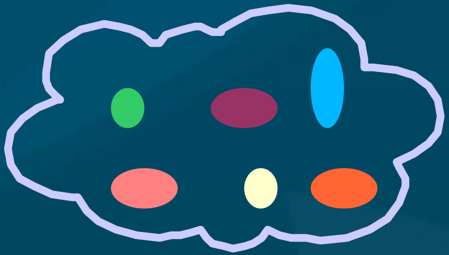
- *Quiz ch1 today*

Announcements

- ACM programming competition:
  - Qualifier rounds @SFU 16Sep and 23Sep
  - World finals in Hawaii if we make it!
  - Free pizza after the qualifier rounds
  - C/C++/Java on Linux (vi/emacs)
  - Our team last year nearly won top prize in CCCU competition
  - Register before Wed 13Sep with Alma: Alma.Barranco@twu.ca

Review from 1.1-1.6

- Tools, toolsmiths
- WADES
- Atomic vs. compound data (examples?)
- Data types (examples?)
  - What's the difference: 5, 5.0, '5', (5), {5}



Quiz ch1

- Get out a blank sheet of paper
- In the top right corner, write
  - Your name
  - Student ID#
  - CMPT14x Quiz 1
  - Today's date (11 Sep 2006)
- Number your answers and provide short answers as best you can
- Closed book, closed notes, closed laptops/calcs

Quiz ch1 (5 questions, 20 marks, 10 minutes)

- Copy this sentence and fill in the blanks:
  - "Computers are t_____, and computer scientists are t___________."
- What are the five steps of top-down problem solving?
  - (okay if you don't get exact words; write the concepts)
- Describe two compound data types.
- What's the difference between 3, 3.0, and "3.0"? Explain.
- What does this evaluate to in Python: 7 / 3

Quiz chapter 1: solutions (#1-2)

- "Computers are tools, and computer scientists are toolsmiths." (2) (2)
- Five steps of top-down problem solving: (5)
  - Write everything down
  - Apprehend the problem
  - Design a solution
  - Execute your plan
  - Scrutinize the results

Quiz chapter 1: solutions (#3-5)

- Compound types: set, tuple (2+2)
  - Also ok: aggregate, array, list, dictionary, hash
- 3, 3.0, “3.0”: difference is type: (2)
  - 3 is integer type (int) (1)
  - 3.0 is float type (a.k.a. Real) (1)
  - “3.0” is string type (str) (1)
- 7 / 3 >>> 2 (2)
  - (integer division)

What's on for today (§1.6.5 - §2.1)

- Variables and constants
- Expressions and precedence
- Logical operators
- Hardware abstractions
- Software abstractions: levels of translation
- Control/structure abstractions
- Pseudocode
- Library functions

Variables and constants

- A constant's value remains fixed: e.g., π, e, 2
- A variable's value may change: e.g., x, numberOfApples
- We can assign new values to variables
  - numberOfApples = 12
  - numberOfApples = numberOfApples – 1
- But not to constants
  - π = 3.0 (don't want to do this!)
- In Python, there is no way to force a name to be constant
  - Convention: use ALLCAPS for names that are intended to be constant

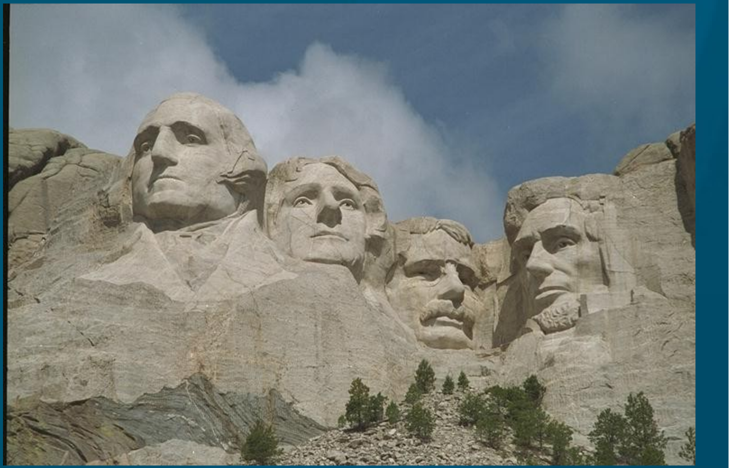
Expressions

- A combination of data items with appropriate operators is called an expression
- Expressions are evaluated to obtain a single numeric result
  - 15 + 9 + 11 + 2 ------evaluation--->>> 37
- Operators may evaluate to a different type than their operands:
  - 22.1 > 15.0:
    What is the type of the operands?
    What is the type of the result?

Logical operators

- Logical operators are operators on the bool type:
  - GodLovesMe = True
  - ILoveGod = False
- not: flips True to False and vice-versa
  - not GodLovesMe >>> False
- and: evaluates to True if both operands are True
  - GodLovesMe and ILoveGod >>> False
- or: evaluates to True if at least one operand is True
  - GodLovesMe or ILoveGod >>> True

Operator Precedence



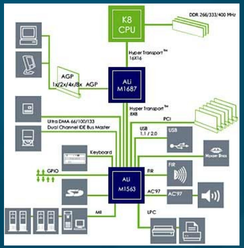
- How would you evaluate this?
  - 5 + 4 * 2
  - (5 + 4) * 2 >>> 18: Addition first
  - 5 + (4 * 2) >>> 13: Multiplication first
- Precedence is a convention for which operators get evaluated first (higher precedence)
  - Usually multiplication has higher precedence than addition
- When in doubt, use parentheses!

Expression compatibility

- 5 + True doesn't make sense: incompatible types
- What about 5(int) + 2.3(float)?
  - Works because the two types are expression compatible
- The "+" operator is overloaded:
  - It works for multiple types: both int and float
- It turns out that in Python, 5+True does evaluate:
  - 5+True >>> 6
    (interprets True as 1 and False as 0)

Hardware abstractions

- Generally, most computers have these basic hardware components:
  - Input
  - Memory
  - Processing
  - Control
  - Output



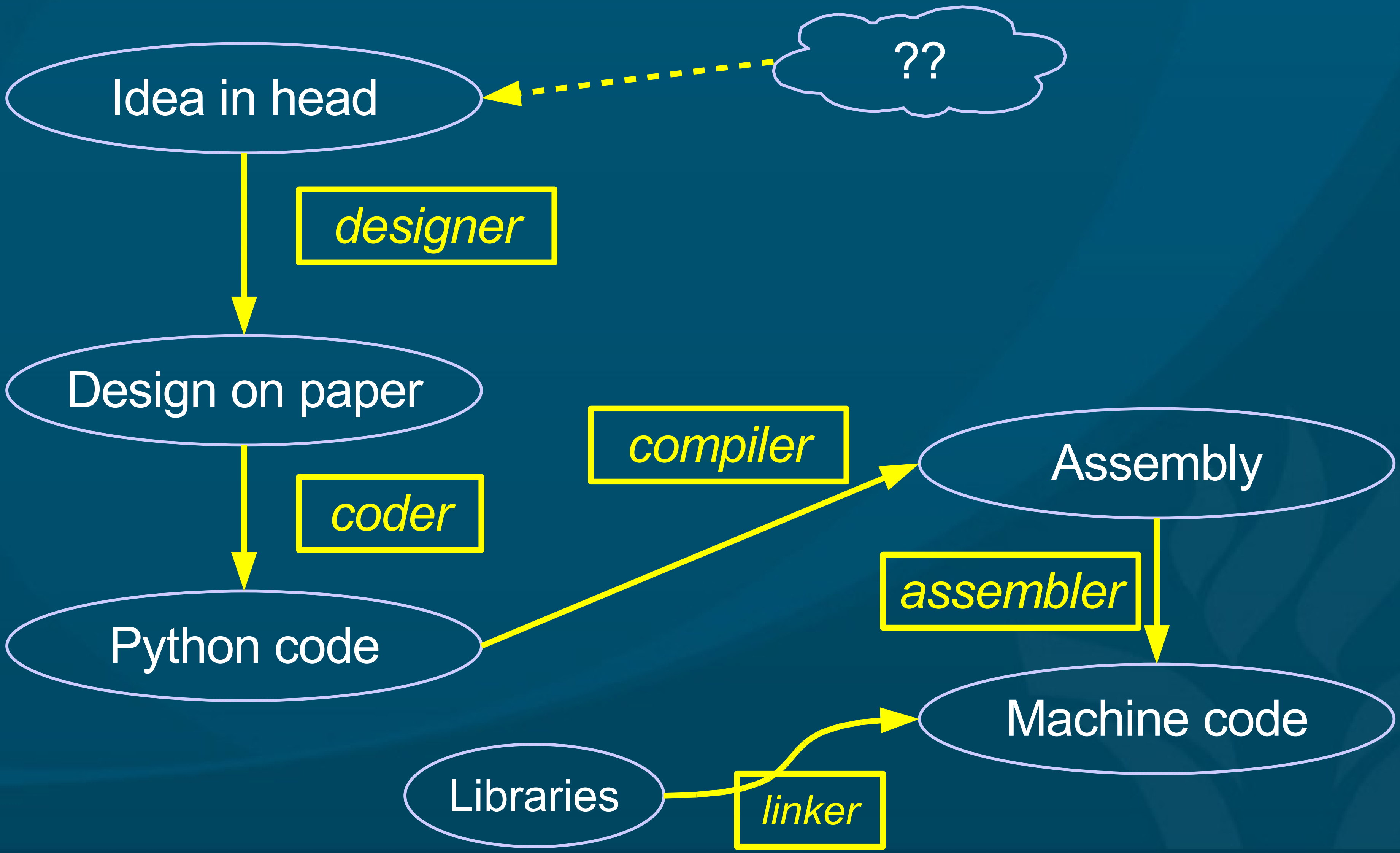
- Together with the software, the environment presented to the computer user by these is the virtual machine

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.

Python

Programming is translation



Control abstractions

- Sequence: first do this; then do that
- Selection (branch): IF ... THEN ... ELSE ...
- Repetition (loop): WHILE ... DO ....
- Composition (subroutine): call a function
- Parallelism: do all these at the same time

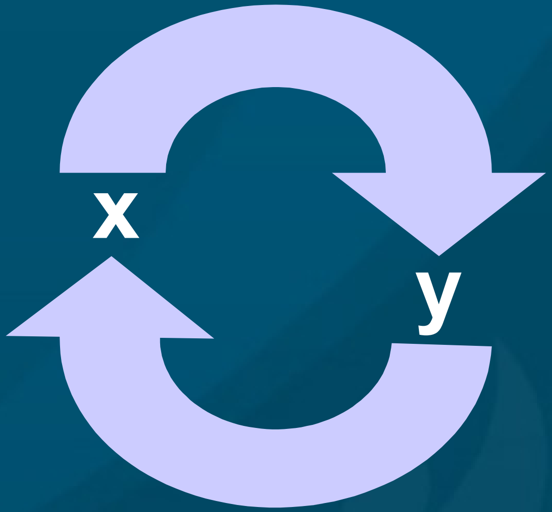
- These are the basic building blocks of program control and structure

Pseudocode

- Pseudocode is sketching out your design
  - General enough to not get tied up in details
  - Specific enough to translate into code
- Use the five control abstractions
- Usually several iterations of pseudocode, getting less abstract and closer to real code
- Don't worry about syntax; worry about semantics
  - Repetition can be done with WHILE ... DO ... or LOOP ... UNTIL ....:
  - Similar semantics; different syntax

Example pseudocode: swap

- Problem: swap the values of x and y
- Initial solution:
  - x <--- y
  - y <--- x
- Will this work?
- Try again:
  - temp <--- x
  - x <--- y
  - y <--- temp



Example pseudocode: add 1..20

- Problem: add the integers between 1 and 20
- Initial solution:
  - Initialize sum to 0
  - Initialize counter to 1
  - Repeat:
    - Add counter to sum
    - Add one to counter
  - Until counter = 20
- Will this work?

Example: add 1..20 (second try)

- Try again:
  - Initialize sum to 0
  - Initialize counter to 1
  - Repeat:
    - Add counter to sum
    - Add one to counter
  - Until counter = 21

- Alternate version:
  - Initialize sum to 0
  - Initialize counter to 1
  - While counter <21, repeat:
    - Add counter to sum
    - Add one to counter

- Same semantics, different syntax
- Top-of-loop test vs. bottom-of-loop test

Pseudocode: you try (group effort!)

- Problem: print the largest of a sequence of numbers
  - Set Curmax to negative infinity
  - Loop:
    - Select next number:
    - See if it's bigger than curmax:
    - If it is, set it as new curmax
    - Repeat until no more numbers
  - Print curmax

Importing library functions

- Library functions are building blocks:
  - Tools that others wrote that you can use
- Functions are grouped into libraries:
  - If you want to use a pre-written function, you need to specify which library to import it from

```
import math
math.sqrt( 2 )          >>>1.4142135623730951
math.pow( 3, 5 )            >>>243.0
math.pi                     >>>3.1415926535897931
```

Review of today (1.8-2.1)

- Expressions and precedence
- Logical operators
- Five abstract components of hardware
- Software: instructions, languages, programs, operating system
- Designer -> coder -> compiler -> assembler + linker
- Five control/structure abstractions of programs
- Pseudocode
- Importing library functions

Writeups for Labs 1-2 *(L1 due next wk)*

- Full writeups required starting with Lab3
- Labs1-2 can have short writeup:
  - Design (10 marks)
    - Name, student#, CMPT14x, lab section, Lab#1, date
    - Statement of the problem
    - Discussion of solution strategy
  - Code (30 marks)
    - Name, etc. again in code header
    - Well-commented code, formatted and indented
  - Output (10 marks)
    - A couple runs with different input

TODO items

- Go to Neu9 computer lab:
  - Make sure you can login
  - Python/IDLE intro on course www (due Wed)
    - Nothing to hand in on this intro
- Homework due next class (Wed):
  - §1.11 # 25, 31, 40
- Reading: through §2.2 for Wed
- Lab1 due next week MTW (in lab section)
- Remember your quiet time journals