# Review ch1-5

3 Oct 2007
CMPT14x
Dr. Sean Ho
Trinity Western University

## Ch1-5 overview

- Ch1: Problem solving
  - Top-down, WADES
- Ch2: Your first Python program
  - Modules, variables, expressions, type
- Ch3: Program Structure
  - Sequences, if, loops
- Ch4: Procedures
  - Parameters, return values, recursion
- Ch5: Arrays and lists
  - Creating, iterating, operations

## Ch1: Problem solving

- Computing scientists as toolsmiths
- Top-down vs. bottom-up; WADES
- Client --> Designer --> Implementer
  - Requirements doc, Design spec, Code
- Abstract data types
  - Atomic vs. compound
  - What's the difference: 5, 5.0, '5', (5), {5}
- 5 hardware abstractions
- 5 control/flow abstractions

## Review §1.5-1.7

- Operators, operands, ADTs, implementations
- Variables vs. constants
- Logical operators: not, and, or

```
=       NOT      *
AND     /            <
                 -
+          OR
```

## Review: §1.8-2.1

- Expressions and precedence
  - http://docs.python.org/ref/summary.html
- Five abstract components of hardware
- Software: instructions, languages, programs, OS
- Designer -> coder -> compiler -> assembler/linker
- Five control/structure abstractions of programs
- Pseudocode
- Importing library functions

## Review: §2.2, 2.5, 2.11

- Components of a baby Python program
- Modules
- Library tools (what are some we know already?)
- Literals, identifiers and reserved words (examples?)
- Strings, quoting, newlines
- Statically-typed vs. dynamically-typed
- Declaring and initializing variables
  - (what is needed in C? In Python?)
- Keyboard input:
  - input(), raw_input()

## Review: §2.3-2.4

- Documentation
  - External documentation: design, manuals
  - Internal documentation:
    - Comments
    - Docstrings
  - Preconditions / postconditions
- Style guidelines

## Review: §2.7-2.10

- Expressions, operators, operands
  - Binary arithmetic: + - * / % // **
  - Comparison: == < > <= => != <> is, is not
  - Boolean: and or not (shortcut semantics)
- Type conversions
- Precedence rules
- Formatted output
  - %d, %f, %s

## Sample quiz ch2

- Name the five software control/flow abstractions
- Evaluate the following Python expressions:
  - **3.0 >= 1 and 3.0 <= 10**
  - **True and (3 <> 5.7)**
  - **not False or (12 % 0)**
  - **3 + 32 // 5.0**
- Show the output of this Python code:
  - **print “I have %04d %s.” % (23.7, “apples”)**
- Assume that the variable numApples has integer type. Write a line of pseudocode that would work in a dynamically typed language like Python but would fail in a statically typed language like C.

## Review: §3.1-3.8

- Selection: if, if..else.., if..elif..else
- Loops: while
- Sentinel variables
- Loop counters
- Using mathematical closed forms instead of loops
- abs(), += etc., string.capitalize()

## Review: §3.4-3.10, 5.4

- String concatenation (+), repetition (*)
- Qualified import
- while loops: continue, break, else
- Common mistakes in loops
- for loops
- range()

# Sample quiz ch3

- Evaluate as Python, or explain the error: [9]
  - (2**4 > 10) or (7 % 3 == 2)
  - 9.0 // 2 == 4.5 and 9 / 0 != 0
  - 'y' + 3 * 'a' + 'y'
- Show the output of this loop: [5]

```
for x in range(4):
    for y in range(4):
        if x == y:
            break
        print "(%d, %d)" % (x, y),
```

- Write pseudocode to convert inches to cm or vice versa, depending on the user's choice [6]

## Review: §4.1-4.3

- Procedures (functions, subroutines)
  - No parameters
  - With parameters
  - Formal vs. actual parameters
  - Scope
  - Global variables (why not to use them)
  - Call-by-value vs call-by-reference

## Review: §5.1-5.3

- Call stack, backtrace
- Abstract Data Types
  - Type hierarchy
- Enumerations
- Arrays
- Lists

## Review: §5.5-5.10, Py ch8

- Python lists vs. M2/C arrays
- Lists as function parameters
- Multidimensional arrays/lists
- Python-specific list operations
  - Membership (in)
  - Concatenate (+), repeat (*)
  - Delete (del), slice ([s:e])
  - Aliasing vs. copying lists

# Sample quiz ch4-5

- Name two standard container (aggregate) types in Python.
- Name two operations/functions/properties that Python lists have that M2/C arrays do not.
- Write a Python function

  `create_matrix(n_rows, n_cols)`

  that returns a new matrix of the specified size.
  - Contents of the matrix don't matter
  - Docstring required!
  - Partial credit for pseudocode

## TODO

- Midterm ch1-5 this Friday!
  - Includes material in texts (M2 and Py) not covered in class!
  - Expect questions similar to quizzes
  - Bring blank sheets of paper
  - Closed book/notes/laptop/phone/calc
- Lab04 next Wed:
  - M2 ch5 # 26 / 32 / 38 / 39
- HW04 next Fri:
  - Py §8.3 #1
  - Py §10.7 #1