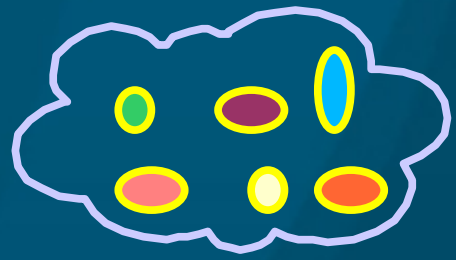
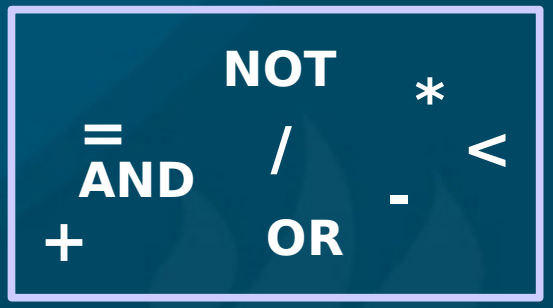
Writing Programs: Pseudocode, Documentation

21 Sep 2010
CMPT140
Dr. Sean Ho
Trinity Western University

Review



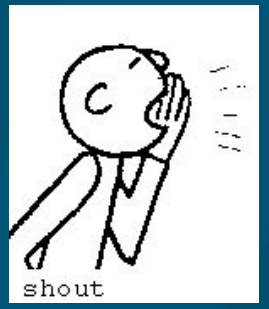
- Data types (examples?)
  - Contrast: 5, 5.0, '5', “5”, (5), {5}
- Operators, operands, ADTs, implementations
- Variables vs. constants
- Logical operators: not, and, or
- Operator precedence
- Expression compatibility (what types?)
- Pseudocode



Static vs. dynamic typing

- All variables have a type: int, float, str, bool, ...
- Some languages (C, Java, M2): statically typed:
  - Must declare variable type ahead of time:
    - **x, y: REAL;**
    - **int numApples;**
  - Can't change the type or assign a value of a different type:
    - **x := “Hello, World”; (* won't work! *)**
- But Python is dynamically typed:
    - **x = 5.0**
    - **x = True # works in Python**

Declaring vs. initializing



- This is only necessary for statically-typed languages:
  - Declare a variable to tell the compiler the type of the variable:
    - **VAR numApples : CARDINAL; (* M2 *)**
  - Its value is undefined until it is initialized:
    - **BEGIN**
      - **numApples := 5; (* M2 *)**
- In a dynamically-typed language like Python, just initialize the variable:
    - **numApples = 5 # Python**

Type conversions

- Python is dynamically typed, so operators can do implicit type conversions to their operands:
  - 2 *(int)* + 3.5 *(float)* → 5.5 *(float)*
    - (+) op converts 2 *(int)* to 2.0 *(float)*
- You can also manually convert types:
  - int(2.7) → 2
  - int(True) → 1
- but this returns an error:
  - int("apples!") → *ValueError*

Keyboard input

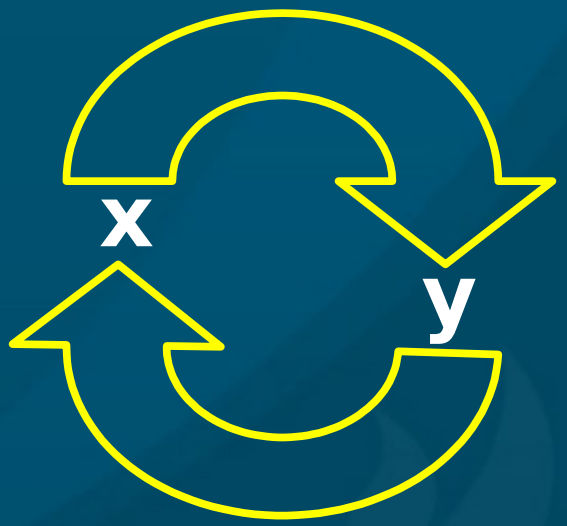
- Use input() to get a value from the user:
  - balance_str = input("Opening balance? ")
  - The argument is the prompt string
    - Note trailing space in the prompt
  - Returns a string: Python doesn't interpret
- Use type conversion to interpret:
  - balance = float(balance_str)
- Or do it all in one line:
  - balance = float(input("Opening balance? "))

Pseudocode

- Pseudocode is sketching out your design
  - General enough: not tangled in details
  - Specific enough: can translate into code
- Use the five control abstractions
- Usually several iterations of pseudocode, getting less abstract and closer to real code
- Don't worry on syntax; focus on semantics
  - e.g., repetition can be done with WHILE ... DO ..., or LOOP ... UNTIL ..., etc.
  - 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: 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 *(2nd 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
  - Initialize max to the first number in the sequence
  - while there are more numbers:
    - get the next number, store in num
    - if num > max:
      - **update max with num**
  - print max

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

- Short writeup (full writeups required starting with Lab3)
  - Design (10 marks)
    - Name, CMPT140, 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
    - Clear, well-chosen identifiers (variable names)
  - Output (10 marks)
    - A couple runs with different input

TODO items

- Lab1 due Thu 10pm. Upload to myCourses:
  - Lab write-up (simplified form)
  - Code (*.py)
  - Screenshots of trial runs (may include within the write-up)
- Quiz2 next Tues in-class (lectures 3-5)
- Lab2 is due next week Thu