# Types, Expressions, Variables

16 Sep 2010
CMPT140
Dr. Sean Ho
Trinity Western University

# Quiz 1 (5min; 10pts)

- Expand and explain: WADES **[3]**
- The end-user client and the design team need to dialogue to agree on a ______ document. **[2]**
- Contrast: syntax vs. semantics **[2]**
- Contrast: producer vs. director **[3]**

# Quiz 1 answers (10pts)

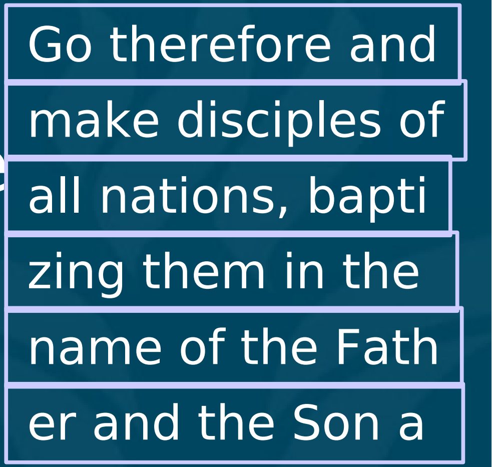
- Expand and explain: WADES **[3]**
  - Write, Apprh'nd, Design, Execute, Scrut'nz
- The end-user client and the design team need to dialogue to agree on a ______ document. **[2]**
  - Requirements
- Contrast: syntax vs. semantics **[2]**
  - Punctuation, spelling, etc. vs. meaning
- Contrast: producer vs. director **[3]**
  - keep on track vs. grand vision

# 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

# Operators care about type

- Operators work on operands:
  - e.g. 3+4: operator is “+”; operands: 3, 4
- Cardinal type: e.g., +, -, *, /, *print*, etc.
- Character type: e.g., *capitalize*, *print*, etc.
  - 'b' / '4' doesn't make sense
- String type: e.g., *reverse*, *print*, etc.
  - *reverse*(1.3) doesn't make sense
- Array-of-strings type: e.g.,
  - Reverse each string in the array
  - Reverse the order of the array



Go therefore and
make disciples of
all nations, bapti
zing them in the
name of the Fath
er and the Son a

# Abstract Data Types

- An Abstract Data Type (ADT) is a set of items w/ common properties and operations
  - e.g., Real ADT: reals w/ +, -, *, /, ...

- Implementation of an ADT:
  - Real-world implementations of ADTs on actual computers have limitations
  - e.g. Can't represent integers bigger than 2,147,483,647 (on a 32-bit machine)
  - e.g. Real (floating-point) numbers can be represented only up to a certain number of significant figures: 1.9999999999 ≠ 2

# Variables and constants

- A constant's value remains fixed: e.g., π, e, 2
- A variable's value may change: x, numApples
- We can assign new values to variables
  - numApples = 12
  - numApples = numApples – 1
- But not to constants
  - π = 3.0 (don't want to do this!)
- In Python, can't 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 take bool operands:
  - GodLovesMe = True
  - ILoveGod = False
- not: flips True to False and vice-versa
  - not GodLovesMe → False
- and: is True if both operands are True
  - GodLovesMe and ILoveGod → False
- or: is 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 is nonsensical: incompatible types
- What about 5 *(int)* + 2.3 *(float)* ?
  - Works because the two types are expression compatible
- The “+” operator is overloaded:
  - Works for multiple types: int and float
- It turns out that in Python, 5+True does work:
  - 5 + True → 6
    (interprets True as 1 and False as 0)

# 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

# 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