

Ch1-7 Review

20 Oct 2008

CMPT14x

Dr. Sean Ho

Trinity Western University

Quiz 04: 10 minutes, 20 points

- Contrast **aliasing** a list with **copying** a list.
 - Write **Python** code to demonstrate the difference
- Contrast a **header** (**DEF**) file with an **implementation** (**IMP**) file.
- Why should we use **accessor** (set/get) functions in an ADT?
- Write a Python function **matrix(n_rows, n_cols)** to create and return a **2D list** with the given number of rows/cols.
 - Contents of the matrix don't matter

Quiz 04 (ch5-6): answers #1

- Contrast **aliasing** a list with **copying** a list.
 - **Aliasing**: another name for the same list; modifying elements of the alias also modifies elements of the original list
 - **Copying**: a separate data structure with the same contents

```
origList = [ 1, 2, 3 ]
aliasList = origList
copyList = origList[:]
aliasList[0] = 5
copyList[1] = 7
origList[0] # is 5
```

Quiz 04 (ch5-6): answers #2-3

- Contrast a **header (DEF)** file with an **implementation (IMP)** file.
 - Header: public interface, doesn't define bodies of functions
 - Implementation: contains bodies of the functions
- Why should we use **accessor (set/get)** functions in an ADT?
 - Hide implementation details from user
 - Maintain the “illusion” of the ADT
 - Ease future upgrades of internal implementation

Quiz 04 (ch5-6): answers #4

- Write a Python function `matrix(n_rows, n_cols)` to create and return a **2D list** with the given number of rows/cols.

```
def create_matrix(n_rows, n_cols):  
    matrix = range(n_rows)  
    for row in range(n_rows):  
        matrix[row] = range(n_cols)  
    return matrix
```

Today: Chapters 1-7 Review

- Ch1: Problem-solving
- Ch2: Your first program
- Ch3: Program structure
- Ch4: Procedures/functions
- Ch5: Arrays/lists
- Ch6: Library modules
- Ch7: Applications

Ch1: Problem solving

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

Ch2: A basic Python program

- Components of a baby Python program
- Literals, identifiers and reserved words (examples?)
- Strings, quoting, newlines
- Statically-typed vs. dynamically-typed
- Declaring and initializing variables
- Keyboard input: `input()`, `raw_input()`
- Expressions, operators, and precedence rules
- Formatted output: `%d`, `%f`, `%s`

Ch3: Basic Program Structure

- Statement **sequences**
- **Selection** (if, else, elif)
- Repetition/**loops** (while, for)
 - Top-of-loop vs. bottom-of-loop testing
 - Sentinel variables
 - continue, break, else
- Sequence **concatenation** (+) and **repetition** (*)
 - ◆ Works on strings, lists, tuples

Ch4: Functions

- **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**
 - ◆ Python is call-by-**object**, which is like call-by-**value** for **immutable** types and call-by-**reference** for **mutable** types

Ch5 (+Py ch8): Arrays and Lists

- Call stack, backtrace
- 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

Python type hierarchy (partial)

- **Atomic** types

- Numbers

- ◆ Integers (int, long, bool): **5, 500000L, True**
- ◆ Reals (float) (only double-precision): **5.0**
- ◆ Complex numbers (complex): **5+2j**

- Container (**aggregate**) types

- Immutable sequences

- ◆ Strings (str): **"Hello"**
- ◆ Tuples (tuple): **(2, 5.0, "hi")**

- Mutable sequences

- ◆ Lists (list): **[2, 5.0, "hi"]**

- Mappings

- ◆ Dictionaries (dict): **{"apple": 5, "orange": 8}**

Ch6: Libraries

- (skip sections on standard I/O)
- Python standard **math** library
- Libraries: interface (DEF) vs implementation (IMP)
- **Accessor** (set/get) functions

Ch7: Applications

- Null-terminated strings; **lexical** sorting
- **fractions.py** ADT library:
 - Set/get functions to hide tuple implementation
- **substitution.py** cipher library:
 - How it works, encode/decode
- **pseudorandom.py** RNG library:
 - Seed, iterative process
 - ◆ (Understand concepts enough to code it)
 - Testing via histograms