§9.9-9.13 and Py ch12: Records and Objects

- *Announcements*

27 Oct 2006
CMPT14x
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Welcome to CMPT 145!

- We're now in the second portion of the semester
- You have the basics of programming now!
- Advanced topics: data structures, algorithms, object-oriented design, etc.

- Paper on computing and society:
  - 1-paragraph topic abstract due Mon 13Nov
  - 5-page paper due last day of class Wed 6Dec

Essay / Paper

- Computing scientist as **G**odly **C**hristian **L**eader:
  - Not just knowledge about tools, but
  - Wisdom of how to use tools
    - To serve others and
    - To give glory to God
- Write a short essay on a topic of your choosing about computers and society:
    - Approx 5 pages typed double-spaced 12pt 1in margins
    - Submit half-page topic by Mon 13Nov
    - Paper due last day of class (Wed 6Dec)
      - Electronic submission ok (email, eCourses)

Sample paper topics

- Censorship and free speech
  - Pornography, gambling, hate groups, etc.
- Violence in video games (Columbine etc.)
- Privacy: online banking, ID theft, etc.
- Blogs: effect on politics, social interaction, etc.
- File sharing: Napster, Gnutella, etc.
- Artificial intelligence: the nature of sentience
- Online dating (e.g. eHarmony): pros/cons
- Equity of access / rural digital divide
- ... come up with your own topic!

Tips for essay writing

- Your essay should be a position paper:
  - The topic should have at least two sides (e.g. pro/con)
  - You should state (in the introductory paragraph) what your position is (thesis)
  - You should have at least 2-3 points, each, both for and against your position
    - It is not necessary to rebut every point that contradicts your position:
    - Be honest about the faults/limitations of your thesis
  - Summary intro/conclusion paragraphs
  - Proper English (spelling, grammar) is important!

Records

- Say we want to create a student info database:
  - First name
  - Last name
  - Student ID #
  - Year
- How do we store this?
  - Four separate lists:
    - firstNames = ['Tom', 'Alan', 'Yuri', 'Megan', ...]
    - studentID = [38, 28, 10, 49, ...]
  - Or one list of student records

User-defined types

- A record is a user-defined aggregate type:
  - Define a StudentRecord type as:
    - First name (string)
    - Last name (string)
    - Student ID (integer)
    - Year (integer between 1 and 4)
- Then we can store the whole database in one list, where each entry of the list has type StudentRecord.

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}

Records in M2

- We define a record type in M2 like this:

```
TYPE
    StudentRecord =
        RECORD
            firstname : ARRAY [0 .. 255] OF CHAR;
            lastname : ARRAY [0 .. 255] OF CHAR;
            ID : CARDINAL;
            year : CARDINAL;
        END;
```

- Declare and initialize a new student:

```
VAR
    student1 : StudentRecord;
student1.firstname := “Joe”;
```

Records in Python: Classes

- In Python, classes are user-defined types:
  - Class StudentRecord:
    - firstName = ""
    - lastName = ""
    - ID = 0
    - year = 0
  - Instantiate a new object of type StudentRecord:
    - student1 = StudentRecord()
    - student1.firstName = 'Tom'
- student1 is an instance of the class StudentRecord
  - "x is a variable of type int"

Objects are mutable: copy vs. alias

- Objects are mutable:
  - `student1.ID = 25`
  - `student1.ID = 38`
- This means assignment is just aliasing:
  - `student2 = student1`
  - `student2.ID = 50  # affects student1.ID`
- To make a separate copy, use copy.deepcopy():
  - `import copy`
  - `student2 = copy.deepcopy(student1)`
- Or create a new instance, and copy values:
  - `student2 = StudentRecord()`
  - `student2.ID = student1.ID`

Using 'id' to look at aliases

- We can check whether two names are aliases or separate copies by using the Python built-in 'id':
  - `id(student1)` `# 11563216`
  - `student2 = student1` `# alias`
  - `id(student2)` `# 11563216`
  - `student2 = copy.deepcopy(student1)` `# copy`
  - `id(student2)` `# 18493888`

Creating a list of objects

- Our student db is a list of StudentRecords
- Because of aliasing, we can't use this shortcut:
  - student = StudentRecord()
  - studentDB = [student] * 35
    - A list of 35 aliases to the same object!
- Use a for loop to create separate objects:
  - **studentDB = [0] * 35**
  - **for idx in range(len(studentDB)):**
    - **studentDB[idx] = StudentRecord()**

TODO

- No lab due next week!
- Lab07 due 6/7/8 Nov:
  - Ch9 # (38+39) / (40+41) / 46
- HW08 due next Mon:
  - Py ch12 # 3, 4
- Paper topic by Mon 13Nov