§9.0-9.9: Sets and Records

29 Oct 2007
CMPT14x
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

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 Fri 9Nov
    - Paper due last week of class (Mon 3Dec)
      - Electronic submission (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!

Set operations

- A set is an unordered collection of items
- Set membership: test if an item is in the set
- Set union: A ∪ B:
  - Anything that's in either A or B
- Set intersection: A ∩ B:
  - Those items which are in both A and B
- Set difference: A – B (or A \ B):
  - Those in A but not in B
- Set symmetric difference: A ^ B:
  - Those in exactly one of A or B

Sets in Python

- Python has a built-in type for sets (as does M2):
  - Instantiate with any iterable (e.g., a list):

    ```
    bagOfApples = set( [ 'Fuji', 'Gala', 'Red Delicious' ] )
    ```

  - Add an apple to the bag:

    ```
    bagOfApples.add( 'Rome' )
    ```

  - Remove an existing apple from the bag:

    ```
    bagOfApples.remove( 'Rome' )
    ```

  - Check if an apple is in the bag:

    ```
    if 'Fuji' in bagofApples:
    ```

- See Python documentation:
  http://docs.python.org/lib/types-set.html

Python set operators

- Operators for Python sets:
  - Union of two sets: .union() or |

    bagOfApples.union(yourApples)

    bagOfApples | yourApples
  - Intersection of two sets: .intersection() or &
  - Difference of two sets: .difference() or –
  - Symmetric difference: .symmetric_difference() or ^
  - Subset: .issubset() or <=
    - A <= B: everything in A is also in B
  - Superset: .issuperset() or >=

Bitsets

- Another way to use sets in Python is to use the binary form of an integer to represent flags:
  - e.g., file permissions

```
readFlag = 1 << 2
writeFlag = 1 << 1
execFlag = 1 << 0
myPerms = readFlag | writeFlag# both read/write

if myPerms & readFlag:                # have read perm
```

- myPerms is called a bitset: it is a compact way of representing a set

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.

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 items

- Register for CMPT145 if you haven't already
- Quiz05 (ch7-8) on Wed
- Lab06 due Wed: ch7 (choose one):
  - # 22: word search game
  - # 32: graphical analysis of pseudorandom
  - # 37: matrix library
  - # 43: encryption algorithms
- Paper topic due next week Fri 9Nov