

Review: Lectures 10-18 (chs 4-5, 7, 9-12)

16 Nov 2010

CMPT140

Dr. Sean Ho

Trinity Western University

Quiz5 (5min, 10pts)

- In programming, what are **exceptions**?
Why are they useful? [3]
- A **ChurchMember** has
a **name** and **phone** number (both strings).
 - **Design** a Python class for **ChurchMember**:
list all relevant **methods** (including [3]
constructor, set/get, and type conversion)
 - **Define** the class and write the **constructor**.
Make sure the attributes are private. [3]
 - Create an **instance** of your class with the [1]
name, "**Joe Smith**" and phone "**555-1212**".

Quiz5: answers #1-2a

- In programming, what are **exceptions**?
Why are they useful? [3]
 - A way of **terminating** execution of the current context, **without exiting** the program
 - Exceptions are objects that can be **raised**; exec. continues at closest matching **handler**
- **Design** a Python class for **ChurchMember**:
list all relevant **methods** [3]
 - **`__init__`**, **`setName`**, **`setPhone`**, **`getName`**,
`getPhone`, **`__str__`**

Quiz5: answers 2b-2c

- Define the class and write the constructor. Make sure the attributes are private. [3]

```
class ChurchMember:  
    def __init__(self, n="", p=""):  
        self.__name = n  
        self.__phone = p
```

- Create an instance of your class with the name, "Joe Smith" and phone "555-1212". [1]

```
joe = ChurchMember("Joe Smith", "555-1212")
```

Overview: lectures 10-18

- Functions
- OO: using objects, designing classes
- Graphics library (concepts, not specific names)
- Lists
- File I/O and pickle
- Exceptions

Review: Functions

- Why use functions, how to define them
 - Definition vs. invocation
- Formal and actual parameters
 - Call-by-value vs. call-by-reference
- Return values

Review: OO

- **Terminology**: object, class, instance, attribute, method, constructor
- **Using** classes: call **constructor** → get **instance**
 - Objects are **mutable** → aliasing, CBR
- **Designing** classes:
 - Class **diagram**, relationships, multiplicity
- **Writing** classes:
 - **Constructor** (with default params)
 - **Private** attributes and **set/get** methods
 - Type **conversion** methods (e.g., **__str__**)

Review: graphics.py

- Concepts, not specific class/method names
- GraphWin
- Point, Circle, Line, Rectangle, Polygon
 - setOutline()
- Text, Entry
 - set/getText()
- getMouse()

Review: Lists

- Compared to C arrays
- Creating lists
- Iterating over lists (for loop)
 - Iterating over multi-dimensional lists
- Lists are mutable → alias / CBR
 - Writing functions which take lists as params
- List operations:
 - len, +, *, in, del, [:]

Review: File I/O

- **Concepts:** file names, current directory, file **handle/object**
 - File **position** pointer: **.seek()**, **.tell()**
 - I/O **streams**, channels, source/sink, **stdin/stdout/stderr**
- **Opening** a file: **open()**, **with**
 - File **modes:** **r**, **w**, **a**, **r+**, ... **b**
- **Reading:** **.read()**, **.readline()**
- **Writing:** **.write()**
- **Serialization:** **pickle.dump(x, f)**, **pickle.load(f)**

Review: Exceptions

- Purpose, definition
- When exceptions are raised
 - Calling a function which raises an exception
- Handling exceptions: try/except
 - Handling a specific exception
 - else, finally
- Naming a caught exception
 - Passing+unpacking auxiliary data