The current topic: Review

✓ Introduction
✓ Object-oriented programming: Python
✓ Functional programming: Scheme
✓ Python GUI programming (Tkinter)
✓ Types and values
✓ Logic programming: Prolog
✓ Syntax and semantics
✓ Exceptions

Announcements

- Lab 3 was due today at 10:30 am.
- Aids allowed for the final exam:
  - One double-sided aid sheet, produced however you like, on standard letter-sized (8.5" x 11") paper.
- Exam period office hours:
  - Monday Dec. 8th, 12:30-1:30, SF3207
  - Wednesday Dec. 10th, 12:30-1:30, SF3207
  - Friday Dec. 12th, 12:30-1:30, SF3207
  - Monday Dec. 15th, 12:30-1:30, SF3207
  - Tuesday Dec. 16th, 11:00-12:00, SF3207
- Lab 3 and Project marking:
  - Marking reports will be emailed (as usual) to your ECF accounts.
  - Watch the Announcements page for re-marking deadlines.

Review

- Disclaimer: This is ***not*** a comprehensive review. Topics that aren't mentioned here may still appear on the final exam.

Object-oriented programming: Python

- Variables:
  - Variables store references, not actual values.
  - Built-in types include:
    - Lists
    - Tuples
    - Dictionaries.
    - Strings.
    - Booleans.
    - Numbers: int, float, complex, long int
- Types:
  - Strongly typed: Type restrictions are enforced.
  - Dynamically typed: Types are determined at runtime; there are no type declarations for variables, parameters, return values, etc.
- Code structure:
  - Indentation is meaningful.

Object-oriented programming: Python

- Containers:
  - Collections of objects.
  - Sequences are containers that have some kind of ordering.
  - Mutable vs. immutable.
- Lists:
  - Mutable sequences.
  - Slicing: getting a portion of a list.
  - Splicing: assigning to a slice.
    - May cause the list to grow or shrink.
- Tuples:
  - Immutable sequences.
- Strings:
  - Immutable sequences where each element is a single character.

Object-oriented programming: Python

- Dictionaries
  - Mutable.
  - Not a sequence.
  - Set of key-value pairs.
- Loops:
  - While loops.
  - For loops.
    - Using the range function to make a list of numbers.
- Classes:
  - Inheritance.
  - Constructors.
  - Instance methods and variables.
  - Class variables.
  - Static and class methods.
  - Name mangling.
  - Operator overloading.

Object-oriented programming: Python

- Exceptions:
  - Raising.
  - Catching.
  - Defining.
- Parameters and arguments:
  - Keyword vs non-keyword
  - Mandatory vs optional parameters
- Regular expressions.
- List comprehensions.
  e.g. `T = [2*x for x in range(4)]`
- Iteration:
  - How this relates to `__getitem__()` and `IndexError`.

Object-oriented programming: Python

- Working with files.
- Modules:
  - Importing modules.
  - Getting short-form naming.

Object-oriented programming: Python

- An example:

```
class A:
   y = 1
   def __init__(self):
      self.y += 2

b = A()
c = A()

b.y  # Value is:
c.y  # Value is:
A.y  # Value is:
```


Functional programming: Scheme

- Functions as first-class values.
- List operations:
  - car
  - cdr
  - cons
  - append
- Other operations:
  - Numeric (e.g. +, –, *, /)
  - Comparison (e.g. >, <, <=, >=, =)
  - Type-checking (e.g. number?, symbol?, list?)
  - Boolean (e.g and, or, not)
- Defining functions.
- Conditional execution: if, cond

Functional programming: Scheme

- Efficiency:
  - let, let*
  - helper functions
  - using an accumulator
  - tail recursion
- Lambda expressions.
- Higher-order functions:
  - functions as parameters
  - functions as return values
- Built-in higher-order functions:
  - map
  - eval
  - apply
  - reduce (not built-in in mzscheme, but built-in in some other Schemes)

Functional programming: Scheme

- Trees:
  - representing trees
  - working with BSTs
- Mutual recursion.

Python GUI programming: Tkinter

- The event loop.
- Creating a root window.
  - And creating additional windows.
- Widgets:
  - creating
  - arranging
- Event-handling:
  - creating callback functions
  - setting the callback function for a particular event
  - Canvas event objects

Types and values

- Attributes of a variable:
  - static attributes vs dynamic attributes
  - name
  - memory address
  - type
  - scope
  - lifetime
- Referencing environment:
  - set of names that can be used at a particular point
- Referencing environment for functions passed as parameters:
  - shallow binding: names that can be accessed depend on where function is called
  - deep binding: names that can be accessed depend on where function is defined

Logic programming: Prolog

- Prolog statements:
  - Facts.
  - Rules.
  - Queries.
- Answering queries:
  - Unification.
  - Resolution.
  - Backtracking.
- Working with lists.
- Math.
- Structures:
  - No structural difference between queries and data.

Logic programming: Prolog

- Trees:
  - representing trees
  - working with BSTs
- Cut:
  - what cut does
  - avoiding wrong answers
  - avoiding duplicate answers
  - avoiding unnecessary work
  - green vs red
- Negation:
  - what negation in Prolog really means
  - using negation safely

Syntax and semantics

- Languages:
  - syntax vs semantics
- Using BNF and EBNF to specify syntax.
- Parsing:
  - following a derivation sequence to produce a parse tree
- Generating low-level code from a parse tree.
- Translation:
  - lexical analysis
  - parsing
  - code generation

Exceptions

- What we gain by using exceptions
- Exceptions in Java:
  - Structured.
  - Strict.
  - Checked vs unchecked exceptions.
- Program design with exceptions.