21-0: Python

We spent quite a bit of time learning Python
- Control structures
- Data structures
- Syntax
- Use of built-in modules

21-1: Programming Paradigms

- We saw three different styles of programming in this class
  - Object-oriented
  - Functional
  - Declarative

21-2: Features of “Pure” OO Languages

- Dynamic typing
- Late binding
- Inheritance
- Encapsulation
- Polymorphism

21-3: Functional Programming

- Functional programming focuses on the development of functions
  - Black boxes that can take inputs and return an output
  - No alteration of inputs
  - No side effects
- Functions are “first class” objects
- Purely functional languages:
  - Lisp, Scheme, ML
  - Python has a great deal of support for functional programming

21-4: Example: Reversing a list

- Imperative approach:
  ```python
  def reverse (l) :
    temp = []
    for item in l :
      temp = [item] + temp
    return temp
  ```
- Emphasis is on building up a result and returning it
21-5: Example: Reversing a list

- Functional approach:
  ```python
def reverse(l):
    if l:
        return reverse(l[1:]) + [l[0]]
    else:
        return []
```
- No assignment statements
- Problem is solved via divide-and-conquer

21-6: Functional Programming structures in Python

- map
- reduce
- lambda
- List comprehension

21-7: Declarative Programming

- Declarative programming focuses on relations between data.
- More on what should be done, less on details of how.
- XSLT has a declarative feel
  - Specifies the result for a given input, but not the details of how a document should be transformed.

21-8: Other issues

- We also talked about:
  - Support for debugging/antibugging
  - XML as a specialized data representation language
  - Compilation
  - Parsing and garbage collection

21-9: Antibugging

- Assert
- Exceptions
- Modular design
- Profiling

21-10: XML

- Syntax
- Parsing XML
- XPath
- XML Schema
21-11: Compilation

- Compilers vs Interpreters
- Stages in compilation
  - Lexical analysis
  - Parsing
  - Semantic Analysis
  - Optimization
  - Code generation

21-12: Parsing and GC

- Parsing
  - Grammatical structure
  - Recursive descent
- Garbage Collection
  - Static vs Dynamic Allocation
  - Automatic memory management
  - Fragmentation and compaction
  - Reference Counting
  - Mark and Sweep