

Graduate School of Library and Information Science
Fall 2008

Foundations of Information Processing in Library and Information Science

(LIS 452)

Section LE Wednesday, 6:45 PM–8:45 PM Online

This document is Copyright © 2008 by David Dubin and the Trustees of the University of Illinois. In addition to this syllabus, this course is governed by the rules and guidelines set forth in the [Code of Policies and Regulations Applying to All Students](#) and A Handbook for Graduate Students and Advisers which graduate students receive upon admission to the program. Students should also consult, and take to heart, the Professional Guidelines and Codes of Ethics for Library and Information Science Professionals available from the [GSLIS main office](#).

This syllabus is provided to UIUC students as part of the materials for a particular class. However, it may be copied, redistributed, and modified under the terms of the [Creative Commons Attribution-ShareAlike license](#) (Version 2.0). The text of that license is available on the Worldwide Web at creativecommons.org. Resources that are linked to or referenced from within this syllabus (e.g., readings, outlines, discussions) are not covered by the license, unless specifically labeled as such.

INSTRUCTOR

David Dubin

Office: LIS 330

Office Hours: Thursday, 9:00 AM–12:00, and by appointment

Phone: 217–244–3275 (217–BIG–EARL)

Email: ddubin

Web: <http://people.lis.uiuc.edu/~dubin/>

REQUIRED TEXTS

David Harel. *Computers Ltd.: What They Really Can't Do* (Oxford University Press, 2003).

Bradley N. Miller and David Ranum. *Problem Solving with Algorithms and Data Structures using Python* (Franklin, Beedle and Associates, 2006).

John Zelle. *Python Programming: An Introduction to Computer Science* (Franklin, Beedle and Associates, 2004).

SCOPE AND OBJECTIVES

This course covers the common data processing constructs and computing concepts used in library and information science. The history, strengths and weaknesses of the techniques are evaluated in the context of our discipline. These constructs and techniques form the basis of applications in areas such as bibliographic records management, full text management and multimedia. No prior programming background is assumed.

Objectives

- Present an introduction to computer programming concepts and techniques, geared toward the needs of LIS students and professionals.
- Demonstrate tools to support challenging projects in other LIS classes, and problem solving on the job.
- Provide experience with Python, a high-level language well suited to our needs. Python can be combined with other languages for more advanced development projects than will be covered in this class.

THIS SYLLABUS

The official syllabus for this course is the SGML version that is linked off the class web page. Expressions of the syllabus in other formats are derived from the SGML version. The current SGML version should be consulted to resolve any inconsistencies among other renditions.

ACCESSIBILITY

To insure that disability-related concerns are properly addressed from the beginning of class, students with disabilities who require reasonable accommodations to participate are asked to contact the instructor as early as possible.

REGISTRATION FOR FEWER THAN FOUR CREDIT HOURS

Graduate students are permitted to enroll in this class for two credit hours, rather than the usual four. Those students are expected to complete all assignments with the exception of the term project.

BASIS FOR GRADING AND EVALUATION

The most important standards for success in a class like this one are the educational goals that students bring to the class. Your instructors hope that the activities, assignments, and presentations planned for this semester will be instrumental in your achieving the goals you set for yourself. Each exercise and assignment has been selected to provide an experience that will foster your own learning. Do not think that grading and evaluative feedback are meant as assessments of your success or failure in the class: they are provided as an incentive to engage with the material to the best of your ability, and as a diagnostic, to make sure we find out if you're not getting the benefits that the assignments should provide.

Final grades for graduate students enrolled for four credit hours will be calculated as follows:

- First Programming Assignment: 10%
- Second Programming Assignment: 10%
- Third Programming Assignment: 10%
- Fourth Programming Assignment: 10%
- Midterm Evaluation: 10%

- Term Project: 20%
- Class Participation: 20%
- Final Exam: 10%

Final grades for graduate students enrolled for two credit hours will be calculated as follows:

- First Programming Assignment: 15%
- Second Programming Assignment: 15%
- Third Programming Assignment: 15%
- Fourth Programming Assignment: 15%
- Midterm Evaluation: 10%
- Class Participation: 20%
- Final Exam: 10%

Evaluative and constructive feedback

Students are entitled to both evaluative and constructive feedback on the assignments. Evaluative feedback reports how well a completed assignment satisfied the requirements for a grade. Constructive feedback provides more detailed criticism of the work, and suggestions for improvement.

It is the policy in this class to provide evaluative feedback privately to students when an assignment is returned after grading. However, constructive feedback will be provided in a public forum (usually the class message boards) prior to the time that the assignment is due. The implication of this policy is that students must contribute work for public review far enough in advance of an assignment's due date for the work to be evaluated and commented upon.

On Adapting the Work of Others

Criteria for grading homework assignments include (but are not limited to) creativity and the amount of original work demonstrated in the assignment. However, students are permitted to use and adapt the work of others, provided that the following guidelines are followed:

- Use of other people's material must not infringe the copyright of the original author, nor violate the terms of any licensing agreement. Know and respect the principles of fair use with respect to copyrighted material.
- Students must scrupulously attribute the original source and author of whatever material has been adapted for the assignment. Summarize (e.g. using source code comments) the changes or adaptations that have been made. Make plain how much of the assignment represents original work.

Submitting Assignments to the Instructor

All assignments, including code, sample input and output, and documentation must be submitted in machine readable form. The instructors will discuss detailed requirements for file naming, packaging, and submission for each assignment.

Assignment 1: Variables Branching and Looping

This assignment is due no later than October 2.

Write a Python program and document it with appropriate comments. The program should demonstrate the use of loops, branching logic, and named variables. The exact functionality of the program is up to you. Assignments completed by students in earlier semesters include the following:

- Checks whether a returned book is overdue, and computes fine.
- Checks call numbers against defined ranges and reports classification
- Checks netID to see if a student is enrolled in our class
- Prints a table of book prices converted from US to Canadian or vice versa.
- Reports which day of the week had the greatest number of patron reference requests.
- Parses linux log file and generates a report
- Models rider nausea level on various carnival rides
- Searches for books by key word
- Invites user to guess a number from 1 to 8
- Writes file contents as XML
- Searches for entries in a simple CSV textual database
- Prints possibly mnemonic strings for phone numbers
- Simulates password change tests
- Counts number of days until a particular date

Assignment 2: Subroutines and File I/O

This program is due no later than October 15.

Write a program and document it with appropriate comments. The program should have a main routine and at least two functions or subroutines. The program should read input from a disk file and write output to a disk file. What else the program does, exactly, is up to you. Assignments completed by students in earlier semesters include the following:

- Reads a file, grabs a random winner from a set list and writes winners to a separate file
- A simple address book program
- Stores and tracks student workhours
- Composes a form letter from various data files
- Generates Dublin Core metadata from a source file
- Allows users to add, delete, search, print data in a small book database
- Reads a data file and reports who prefers Diet Coke
- Reads values passed from CGI webpage
- Summarizes voting data by precinct
- Creates pizza order interactively
- Generates a report from video rental data
- Creates a grading curve from homework and exam scores

Assignment 3: Object Class Design

This program is due no later than November 5.

Design an original object class, like those we've discussed. Write an application program that implements and employs the class. Follow these guidelines:

- Decide in detail what attributes ought to be used to implement the class. Document these decisions using comments.
- Design at least two method functions (not including constructors, destructors, and copy methods) for the class. Demonstrate their use in the application.
- Implement the class as a Python module in a separate file from the application program.

Assignments completed by students in earlier semesters include the following:

- Presents a menu of new video releases (class: Movie)
- Simple library circulation program (class: patron, book)
- Simulates bowling game (class: ball)
- Manages state of a chess game (class: board)
- Menu-driven employee evaluation (class: employee)
- Computes frequent flyer miles (class: passenger)
- Parses email messages (class: MailRecord)

Assignment 4: Data structure application

This program is due no later than December 3.

Write a program and document it with appropriate comments. The program should use a stack, queue, tree, or heap to perform some interesting computation. Assignments completed by students in earlier semesters include the following:

- A four-function RPN calculator with functionality and interface similar to the Unix `dc` utility (stack)
- Computes standard deviation of a group of numbers (queue)
- Displays basketball scoring data in an ordered format (queue)
- Computes time conversion between VHS modes (stack)
- Dream date compatibility game (queue)
- Calculates equation of a line from two points (stack)

Midterm Evaluation and Final Exam

All students enrolled in the class are required to complete the midterm evaluation and final exam. The tests will cover the material presented in class and in the reading assignments. The format for the tests will be questions that can be answered in one to three paragraphs.

Term Project

All graduate students enrolled for four credit hours must complete a final project that has been approved by the instructor. The final project represents in-depth investigation of a topic related to data processing, in the form of a computer program and supporting documentation. Proposals for term projects are due no later than October 29. Completed projects are due on December 12 at 5:00 PM, central standard time.

The program must be submitted to the instructor as machine-readable source code, but may be programmed in any language approved by the instructor. Like the program, the documentation must be prepared and submitted in machine-readable form. It should either conform to a standard online format (such as `texinfo` or `man`), or take the form of a report prepared using a declarative markup language (e.g., `DocBook` or `LATEX` format).

Class Participation

The class participation grade is based on consistent attendance, contribution to in-class and/or online discussions, and providing assistance to classmates outside of class. Please alert the instructor if a classmate has been of help to you outside of class.

MILESTONES

To keep on track with the course material and assignments, it is recommended that students master technical competencies according to the following schedule. No grades are assigned for these.

General Technical Milestones

- Participate in a LEEP live session: **week 0 (now)**
- Create a plain text file using a text editor: **week 0 (now)**
- Attach a plain text file to an email message: **week 0 (now)**
- Install Python on your computer: **week 1**
- Install an SSH client (such as PuTTY) on your computer: **week 1**
- Participate in a “double telnet” session: **week 2**
- Configure a Mail User Agent (i.e., an email program) to access your CITES mailbox via IMAP and post email via SMTP: **week 2**
- Create a working directory for your homework and project files in progress. it should be a subdirectory under your secure web directory (courseweb.htmls): **week 2**
- Enable indexing (i.e., directory listing) on your working directory so that you need not provide direct links to your work in progress: **week 2**
- Configure permissions on your working directory so that Linux shell users can enter and list the contents of it and any subdirectories you create. Do not give anyone the power to add, delete or modify files. Enable access (but not listing or modification) for all ancestor directories up to your home directory: **week 3**

Python Milestones

The following milestones are specific to Python programming:

- Execute Python commands interactively: **week 1**
- Create and execute a “Hello World” program in Python: **week 1**
- Manipulate scalar text, numeric, and Boolean data using Python variables: **week 2**
- Use if/then/else logic in simple Python programs: **week 2**
- Manipulate collections of scalar data structures using Python lists, vectors, and dictionaries: **week 3**
- Use for loops and while loops in simple Python programs: **week 3**
- Divide program logic into main and auxiliary procedures/functions: **week 4**
- Read data from text files into Python variables: **week 4**
- Pass parameters to Python functions and obtain return values from them: **week 5**
- Write data to a file from within a Python program: **week 5**
- Load Python modules and import functions and classes from them: **week 6**
- Create object instances from existing classes and execute methods on them: **week 7**
- Create a simple object class, including a constructor: **week 8**
- Extract data from strings using simple regular expressions: **week 8**
- Write instance methods for your object classes: **week 9**
- Use the linked list, queue, and stack modules in your programs: **week 10**
- Use the tree and heap modules in your programs: **week 11**

SEMESTER OUTLINE

Part I: **Librarianship, information processing and information literacy** *August 27*

What does it mean for anyone to be “computer literate?” What learning goals do you have for the semester?

Readings: Syllabus, *MS Computer Literacy Requirements*

Part II: **Computers, programs, and algorithms** *September 3*

Readings: Harel 1; Zelle 1–2

Part III: **Binary representation and file structures** *September 10*

Scalars and collections, typed variables, number systems

Readings: Zelle 3–4

Part IV: **Control structures** *September 17*

Branching and iteration, flow of control

Readings: Zelle 7–8

Part V: **Functional decomposition** *September 24*

Functions and procedures

Readings: Zelle 6; Miller and Ranum 3

Part VI: **On Campus Session** *October 2*

Automata, regular expressions, Linux and X-Windows,

Readings: *Finite state machine reading,*
Regular expression reading

Assignment 1:
variables, branching,
and loops: Due at 5 PM.

Part VII: **Object orientation I** *October 8*

Methods, object identity

Readings: Zelle 5

Part VIII: **Object Orientation II** *October 15*

Inheritance, polymorphism, operator overloading

Readings: Zelle 10, 12

Assignment 2:
subroutines and file
I/O: Due at 5 PM.

Midterm evaluation
distributed: Due October 22, 5 PM.

Part IX: Algorithms and data structures I

October 22

Addresses, pointers, references; dynamic and anonymous data structures: linked lists, stacks, trees

Readings: Zelle 11, Miller and Ranum 2

Part X: Computational complexity I

October 29

logarithmic, linear, polynomial order

Readings: Harel 3; Zelle 13; Miller and Ranum 4

Final project
proposal: Due at 5:00 PM.

Part XI: Algorithms and data structures II

November 5

Heaps, sorting, hashing

Readings: Miller and Ranum 5, 6

Assignment 3: object
class design: Due at 5 PM.

Part XII: Computational complexity II

November 12

N-P Completeness, satisfiability

Readings: Harel 4

Part XIII: Theory of computation I

November 19

Turing machines, nondeterminism, generative grammars

Readings: **Turing Machine reading**

Part XIV: Thanksgiving break

November 26

Part XV: Theory of computation II

December 3

The halting problem, Church's Thesis, Goedel's theorem

Readings: Harel 2

**Assignment 4: data
structure application:** Due at 5 PM.

Part XVI: Wrapup and Evaluation

December 10

Term project: Due December 12 at 5:00 PM, central standard
time

Part XVII: Final Exam

Week of December 19