CS 1510 (810:051)
Introduction to Computing
Syllabus

MEET YOUR INSTRUCTOR

Dr. Mark Fienup, Associate Professor of Computer Science

Dr. Fienup is an Associate Professor of Computer Science. He received his B.A. from the University of Northern Iowa in Mathematics and Chemistry. He earned both his M.S. and Ph.D. in Computer Science at Iowa State University. He has been a faculty member of the Computer Science Department at the University of Northern Iowa since 1985.

Dr. Fienup regularly teaches Introduction to Computing, Data Structures, Computer Organization, and Computer Architecture. His current research activity centers on the Bioinformatics topic of automatics, three-dimensional protein structure prediction.

COURSE OVERVIEW

This course is an introduction to computer programming assuming NO previous programming experience. It is intended for students thinking about majoring or minoring in Computer Science. You will be learning how to write relatively small programs using algorithmic problem solving and procedural abstraction.

Course Objectives

After this course, you should be able to:

- Select appropriate built-in data types (int, float, char, Boolean, array/list, record/struct) to model information for a programming task
- Select and use appropriate programming statements to perform data (numeric and string) processing, selection (if-then, if-then-else, etc.) of code, and iteration (for-loop, while-loop, etc.) of code
- Write "small to medium" sized programs using algorithmic problem solving and functional decomposition in analysis, design, and implementation
- Develop develop appropriate test data to thoroughly test the correctness of a program

COURSE ORGANIZATION

Prerequisites: None

Textbook: Kenneth A. Lambert, *Fundamentals of Python: From First Programs through Data Structures*, 1st Edition, 2010, ISBN-10: 1-4239-0218-1, ISBN-13: 978-1-4239-0218-8**.**

Software Tools: You will need to download and install the free Python development package (version 2.x and NOT version 3.x) from the Python Web-site: http://www.python.org/ Appendix A of the textbook walks you through the installation of Python.

Assignments: Assignments are organized around the chapters 1 through 10 of the textbook which you should cover sequentially. For each chapter, you should follow these steps:

1) Read the chapter thoroughly.

2) Work through the *laboratory assignment* related to the chapter. Each lab will consist of short-answer questions and programming exercises where you will be modifying and writing small sections of Python code. Using a word processor program (Word, Open-Office, etc.), save your short-answer responses to a file in Rich-text format (.rtf). Zip this file and your Python programming files together and submit the .zip file to the appropriate **Lab Submission** link.

3) Take the True/False and Multiple Choice **Chapter Quiz** located in the **Course Content** menu.

4) Check the below schedule to see if the you should complete a **Programming**

Project at the end of the chapter. Programming projects require you to write larger, complete programs so you can improve your program design, implementation, and testing skills.

Examinations:

There will be two proctored exams. A mid-term exam after chapters 1 through section 6.2 and a comprehensive final exam after chapters 6-10. Examination request forms are included at the appropriate places in the **Course Content**.

Course Schedule:

Assignment	Chapter #	Topic	Laboratories	Programming Projects
1	1	Introduction to software, hardware, and Python programming	Lab 1	
2	2	Software development process, data types, and expressions	Lab 2	Project 1
3	3	Control statements: if, for, while	Lab 3	Project 2
4	6.1 to 6.2	Designing with Functions: Top-down design	Lab 4	Project 3
5	4	Strings and text files	Lab 5	Project 4
6	5	Lists and dictionaries	Lab 6 Lab 7	Project 5
Mid-term exam: covering chapters 1 through section 6.2				
7	Rest of 6	Recursive function	Lab 8	
8	7	Simple graphics and image processing	Lab 9	Project 6
9	8	Designing with Classes	Lab 10	Project 7
10	9	Graphical user interfaces	Lab 11	Project 8
11	10	Multithreading, networking, and client/server programming	Lab 12	
Final exam: comprehensive, but focusing on details of chapters 6-10				

GRADING

Grading policy: Course components are weighted as:

Chapter quizzes: 10 %
Laboratories: 15 %
Programming Projects: 25 %
Mid-term exam: 25 %
Final exam: 25 %

Course grades will be assigned based on the following grading scale:

100-90 A
89 - 80 B
79 - 70 C
69 - 60 D
Below 59 F

Plus and minus grades will be assigned for scores within two percentage points from a grade cutoff (e.g., 91.9 to 90 is an A-, and 89.9 to 88 is a B+).

UNI Guided Independent Study requires that you complete all assignments and exams to pass the course.