

## CIS 110 Introduction to Computer Programming

Course Number & Title (A.1)	CIS 110 Introduction to Computer Programming
Credit Units (A.2)	1 CU (1 hour of lecture per week, 2 hours of lab per week, 1 hour of discussion group per week)
Instructor (A.3)	Jean Griffin, Senior Lecturer, <a href="mailto:griffin@seas.upenn.edu">griffin@seas.upenn.edu</a> , 168 Moore Hall, 484-574-4541
Text(s)/Required Materials (A.4)	Course notes on the course web site: <a href="http://www.seas.upenn.edu/~cis110">http://www.seas.upenn.edu/~cis110</a>
Catalog Description (A.5a)	This course is an introduction to the core concepts of computing using the Java programming language. It teaches how real-world problems can be solved computationally using a modern "object oriented" programming language, with an emphasis on data modeling and algorithmic thinking. Topics include basic programming concepts (such as data types, evaluation, conditional logic, iteration, modularity, recursion, arrays, lists, code translation, debugging, and testing), object-oriented principles (such as encapsulation, inheritance, and polymorphism), algorithmic principles (such as pre-conditions and post-conditions, pseudocode, invariants, running time), and software design concepts (such as specifications, top-down design, abstraction, and information hiding). Upon completion of this course, students will be able to reason about how to solve problems computationally, communicate effectively with professional software developers, learn languages other than Java on their own, and succeed in CIS120, CIS240, and other CIS courses that lead toward a major or minor in computer science.
Prerequisites (A.5b)	None
Course Satisfies (A.5c)	[ ] Math [ ] Science [ x ] Engineering [ ] Technical Elective [ ] TBS (check only one, UG curric impact only) <b>Required</b>
Course Web	<a href="http://www.seas.upenn.edu/~cis110/">http://www.seas.upenn.edu/~cis110/</a>
Course Outcomes (A.6a)	<ul style="list-style-type: none"> <li><input type="checkbox"/> Understand that software development involves a thoughtful integration of algorithms and information modeling.</li> <li><input type="checkbox"/> Understand information modeling concepts such as data types, type safety, data structures such as arrays and lists, the stack/heap memory model, encapsulation, inheritance, polymorphism, and abstraction.</li> <li><input type="checkbox"/> Understand algorithmic concepts such as pre-conditions and post-conditions, pseudocode, invariants, running time, recursion, linear processing, divide and conquer, and abstraction.</li> <li><input type="checkbox"/> Understand how to translate problems specified in a variety of formats in to a Java program. Specification formats include: verbal descriptions, state tables, sample program input/output, and Javadocs.</li> <li><input type="checkbox"/> Understand how to communicate effectively with software developers..</li> <li><input type="checkbox"/> Understand how to translate concepts learned in this course to other domains.</li> </ul>
Contribution towards Program Outcomes (A.6b)	a,b,c,d,f,i,j,k
Topics Covered (A.7)	<ul style="list-style-type: none"> <li><input type="checkbox"/> A user-friendly introduction to programming (this semester with Scratch, a graphical, drag and drop, syntax-free programmig language) as an introduction computing concepts such as conditional logic, iteration, variables, and debugging, and events. Note: subsequent topics were covered with Java and pseudocode.</li> <li><input type="checkbox"/> Programming basics: data types, variables, evaluation, operators, operator precedence, conditionals, loops, compilation, binary numbers, boolean logic</li> <li><input type="checkbox"/> Object-oriented programming basics: objects, state, methods, modularity, scope, encapsulation, composition</li> <li><input type="checkbox"/> Memory model: stack memory for method calls, heap memory for objects</li> <li><input type="checkbox"/> Linear algorithms using loops and arrays (e.g. min, max, average, linear search, primality)</li> <li><input type="checkbox"/> Algorithms (pre- and post-conditions, pseudocode, invariants, running time, divide and conquer, abstraction)</li> <li><input type="checkbox"/> Polymorphism with inheritance and interfaces</li> <li><input type="checkbox"/> Recursion</li> <li><input type="checkbox"/> Lists</li> <li><input type="checkbox"/> Graphics and event handling</li> </ul>
Grading Details	40% Homework 40% Exams 10% Lab Participation 10% Discussion Group Participation
Prepared By/Date	Jean Griffin, March 2011