

CIS 120 Programming Languages and Techniques I

Course Number & Title (A.1)	CIS 120 Programming Languages and Techniques I
Credit Units (A.2)	1 CU (3 hours of lecture per week)
Instructor (A.3)	Stephanie Weirich, Associate Professor, sweirich@cis.upenn.edu
Text(s)/Required Materials (A.4)	None. Course notes, handouts, and readings made available on the class web site.
Catalog Description (A.5a)	A fast-paced introduction to the fundamental concepts of programming and software design. This course assumes some previous programming experience, at the level of a high school computer science class or CIS110. (If you got at least 4 in the AP Computer Science A or AB exam, you will do great.) No specific programming language background is assumed: basic experience with any language (for instance Java, C, C++, VB, Python, Perl, or Scheme) is fine. If you have never programmed before, you should take CIS 110 first.
Prerequisites (A.5b)	CIS 110 or equivalent experience
Course Satisfies (A.5c)	[] Math [] Science [x] Engineering [] Technical Elective [] TBS (check only one, UG curric impact only) Required
Course Web	http://www.seas.upenn.edu/~cis120/
Course Outcomes (A.6a)	<ul style="list-style-type: none"> • Ability to use the functional programming paradigm appropriately to implement recursive algorithms (e.g. simple functions over lists and tree-structured data). • Ability to use the object-oriented programming paradigm appropriately for software design. • Ability to use the iterative programming paradigm appropriately for software design (e.g. to implement array-based or stream computations). • Ability to use the event-driven programming paradigm appropriately to implement simple reactive programs (e.g. GUIs). • Ability to implement and use fundamental collection structures such as lists, sets and maps. • Ability to understand and reason about pointers, heap structures, and aliasing. • Knowledge of an abstract model of computation and the ability to reason about the semantics of programs using that model. • Knowledge of the role of types (both static and dynamic) and the notion of subtyping in software design. • Ability to reason about core concepts of the Java language (inheritance, interfaces, overloading, exceptions, etc.). • An understanding of program structure and ability to use third-party libraries and software, such as APIs. • Fluency in the software design process and the ability to understand and implement specifications (both "formal", i.e. types, and "informal", i.e. JavaDocs). • Ability to create test cases and use test-driven development methodologies. • Ability to model real world information as datastructures. • Ability to develop, test, and debug programs at the "medium" scale (up to ~1000 lines of Java.)
Contribution towards Program Outcomes (A.6b)	a,b,c,i,k
Topics Covered (A.7)	<ul style="list-style-type: none"> • test-driven development • data types and data representation • abstraction, interfaces, and modularity • programming patterns (recursion, iteration, events, call-backs, collections, map-reduce, GUIs, ...) • functional programming • how and when to use mutable state • inheritance and object-oriented programming. • Programming in Ocaml • Programming in Java
Grading Details	20% Homework 10% Labs 40% Midterms 30% Final
Prepared By/Date	Stephanie Weirich, May 2011 (based on CIS 120e Fall 2010 syllabus)