

Core Code Concepts

By Jeremy Griffith

The purpose of this document is to provide a basic Computer Science study guide to people who are taking introductory programming courses, studying for basic coding tests, or planning to obtain a software certification. You may be interested in this study guide if you are:

* Taking a Computer Science AP Course
* Studying for the Computer Science AP Test
* Registering for an Introductory Programming College Course
* Brushing up on Concepts for the Major Field Test in Computer Science
* Refreshing Core Skills for a Computer Science Certification Exam

# What this Guide Will Do

This guide will provide you with an overview of all the core competencies needed to be a well-rounded Computer Scientist. The following is breakdown of the entire guide:

1. The Core Concept Checklist
2. Nomenclature
3. Topic Breakdowns

The purpose of the Core Concept Checklist is to give you a way to keep track of all the concepts you’re comfortable with. If any of the terms are new or confusing, you can get a good overview in the nomenclature section. If you still have no idea, each term is covered and broken down into subtopics in the topic breakdown section.

# What this Guide Will **NOT** Do

This guide will not go into a deep dive of any of the subjects described. That’s the entire purpose of my website. All the deep dive material is located there for reference.

# The Core Concept Checklist

# Nomenclature

The purpose of this section is to provide you with a focused list of all the terms you’ll need to understand if you’re studying Computer Science or a related field.

**Algorithm**: a set of instructions for solving a problem

**Application Program Interface (API)**: a collection of exposed interfaces and protocols for the purpose of general reuse

**Concatenation**: the process of combining things together like a pair of strings or several lists

**Constant**: a value that is fixed and cannot be changed

**Comment**: an annotation of a line or section of code

**Compiler**: a software system which converts code to lower-level code

**Documentation**: literature that provides details about a library or tool

**Function**: a set of instructions that can be referenced by a name

**Graphical User Interface**: an interface which allows the use of electronic devices via graphical icons and visual cues

**Integrated Development Environment (IDE)**: a source code editor with automation tools

**Interpreter**: a software system which converts code to lower-level code on-the-fly

**Library**: a collection of implementations for the purpose of general reuse

**Loop**: a repeated sequence of instructions until some condition is met

**Method**: a function that is associated with an object

**Method Overloading**: the ability to define multiple functions in the same scope with the same name

**Method Overriding**: the ability of a subclass to create a more specific version of a method already provided by its superclass

**Polymorphism**: the ability of a variable, method, or object to exist in multiple forms

**Pseudo Code**: an informal or simplified programming language used to describe how a program should execute

**Readability**: the measure of ease of interpretation and understanding of source code

**Recursion**: a method of problem solving which derives the solution from solutions to smaller sections of the same problem

**Variable**: a value that is subject to change

**Version Control**: a system that supports organization of many versions of software