# Programming Focus

The Programming program provides students with the principles of Programming. Areas of study include methodology, algorithms, data structures and object-oriented programming. Java, Python, javascript and C++ are the primary languages taught.

## Introduction

The competencies in this document are designed to clearly state what the student should know and be able to do upon completion of an advanced high school General Programming program. These standards are designed for a two course sequence that prepares the student for technical assessments directly aligned to the standards.

These exit-level competencies are designed for the student to complete all competencies through their completion of a program of study. These competencies are intended to guide curriculum objectives for a program of study.

The standards are organized as follows:

* Competencies are general statements that identify major areas of knowledge, understanding, and the skills students are expected to learn in key subject and career areas by the end of the program.
* Standards follow each content standard. Standards identify the more specific components of each Competency and define the expected abilities of students within each Competency.
* Learning Targets are very specific criteria statements for determining whether a student meets the Standard. Learning Targets may also be used as learning outcomes, which teachers can identify as they plan their program learning objectives.

The crosswalk and alignment section of the document shows where the Learning Targets support the New Hampshire Content Standards. Where correlation with an academic content standard exists, students in the General Programming program perform learning activities that support, either directly or indirectly, achievement of the academic Competencies that are listed.

All students are encouraged to participate in the career and technical student organization (CTSO) that relates to the General Programming program. CTSOs are co-curricular national organizations that directly reinforce learning in the CTE classroom through curriculum resources, competitive events, and leadership development. CTSOs provide students the ability to apply academic and technical knowledge, develop communication and teamwork skills, and cultivate leadership skills to ensure college and career readiness.

The Employability Skills for Career Readiness identifies the “soft skills” needed to be successful in all careers and must be taught as an integrated component of all CTE course sequences. These Competencies are available in a separate document.

The Standards Reference Code is only used to identify or align Learning Targets listed in the Competency to daily lesson plans, curriculum documents, or national standards. The Standards Reference Code is an abbreviated name for the program, and the Competency, Standard, and Learning Targets are referenced in the program Competency. This abbreviated code for identifying competencies uses each of these items. For example, PCS is the Standards Reference Code for General Programming. For Competency 2, Standard 3 and Learning Target 4 the Standards Reference Code would be PCS.2.3.4.

# Algorithms and Programming

Create meaningful and efficient programs including choosing which information to use and how to process and store it, breaking apart large problems into smaller ones, recombining existing solutions, and analyzing different solutions.

## 1.0 Program Development

### 1.1 Program Development

1.1.1 Implement the steps in the System Development Life Cycle (SDLC) (e.g., planning, analysis, design, development, testing, implementation, and maintenance)

1.1.2 Develop program requirements/specifications and a testing plan (e.g., user stories, automated testing, and test procedures)

1.1.3 Apply pseudocode or graphical representations to plan the structure of a program or module (e.g., flowcharting, white boarding, and UML)

1.1.4 Use a program editor to enter and modify code

1.1.5 Identify correct input/output statements

1.1.6 Choose the correct method of assigning input to variables including data sanitization

1.1.7 Choose the correct method of outputting data with formatting and escaping

1.1.8 Differentiate between interpreted and compiled code (e.g., steps necessary to run executable code)

1.1.9 Identify the purpose of a build system (e.g., make, rake, ant, maven, SCons, and grunt)

1.1.10 Name identifiers and formatting code by applying recognized conventions

### 1.2 Documentation

1.2.1 Identify comments

1.2.2 Utilize comments to describe sections of code

1.2.3 Identify the purpose of version control systems (e.g., Git and Mercurial)

1.2.4 Create a new repository

1.2.5 Add, push, and pull source code from repository

1.2.6 Explain branching and its uses

1.2.7 Restore previous versions of code from the repository

1.2.8 Use standard library functions

1.2.9 Find and use third party libraries (e.g., web-based and package managers)

1.2.10 Explain and interact with an Application Program Interface (API)

1.2.11 Apply industry standards in documentation (e.g., self-documenting code; function-level, program-level, and user-level documentation)

### 1.3 Test and Debug

1.3.1 Identify errors in program modules

1.3.2 Identify boundary cases and generate appropriate test data

1.3.3 Perform integration testing including tests within a program to protect execution from bad input or other run-time errors

1.3.4 Categorize, identify, and correct errors in code, including syntax, semantic, logic, and runtime

1.3.5 Perform different methods of debugging (e.g., hand-trace code and real time debugging tools)

1.3.6 Identify runtime errors

1.3.7 Describe error handling strategies

1.3.8 Handle unexpected return values

1.3.9 Handle (catch) runtime errors and take appropriate action

1.3.10 Throw standard exception classes

1.3.11 Develop and throw custom exception classes

### 1.4 Visual Design

1.4.1 Apply W3C standards and style conventions

1.4.2 Construct web pages and applications that are compliant with ADA and sections 504 and 508 standards

1.4.3 Explain the concept of responsive design and applications

1.4.4 Employ graphics methods to create images at specified locations

1.4.1 Choose correct GUI objects for input and output of data to the GUI interface (e.g., text boxes, labels, radio buttons, check boxes, dropdowns, and list boxes)

## 2.0 Programming

### 2.1 Variables

2.1.1 Compare and contrast fundamental data structures and their uses

2.1.2 Declare numeric, Boolean, character, string variables, and float and double

2.1.3 Choose the appropriate data type for a given situation

2.1.4 Identify the correct syntax and usage for constants and variables in a program

2.1.5 Identify the correct syntax and safe functions for operations on strings, including length, substring, and concatenation

2.1.6 Explain complications of storing and manipulating data (i.e., the Big-O notation for analyzing storage and efficiency concerns, etc.)

2.1.7 Research industry relevant programming languages (i.e., Java, JavaScript, Python, etc.)

### 2.2 Lists/Arrays

2.2.1 Demonstrate basic uses of arrays including initialization, storage, and retrieval of values

2.2.1 Distinguish between arrays and hash maps (associative arrays)

2.2.1 Identify techniques for declaring, initializing, and modifying user-defined data types

2.2.1 Create and use two-dimensional arrays

### 2.3 Objects

2.3.1 Make a distinction between an object and a class

2.3.2 Differentiate among inheritance, composition, and class relationships

2.3.3 Instantiate objects from existing classes

2.3.4 Read the state of an object by invoking accessor methods

2.3.5 Change the state of an object by invoking a modifier method

2.3.6 Determine the requirements for constructing new objects by reading the documentation

2.3.7 Create a user-defined class

2.3.8 Create a subclass of an existing class

2.3.9 Identify the use of an abstract class as opposed to an interface

2.3.10 Explain the object-oriented concepts of polymorphism, inheritance, and encapsulation

### 2.4 Boolean Logic

2.4.1 Use the correct syntax for decision statements (e.g., if/else, if, and switch case)

2.4.2 Compare values using relational operators (e.g., =, >, <, >=, <=, and not equal)

2.4.3 Evaluate Boolean expressions (e.g., AND, OR, NOT, NOR, and XOR)

2.4.4 Use the correct nesting for decision structures

### 2.5 Functions/Methods

2.5.1 Demonstrate refactoring techniques to reduce repetitive code and improve maintainability

2.5.2 Demonstrate the use of parameters to pass data into program modules

2.5.3 Demonstrate the use of return values from modules

2.5.4 Write code that creates and calls functions.

2.5.5 Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.

### 2.6 Loops

2.6.1 Identify various types of iteration structure (e.g., while, for, for-each, and recursion)

2.6.2 Identify how loops are controlled (variable conditions and exits)

2.6.3 Use the correct syntax for nested loops

2.6.4 Compute the values of variables involved with nested loops

## 3.0 Algorithms

### 3.1 Mathematical Algorithms

3.1.1 Identify and correctly use arithmetic operations applying the order of operations (precedence) with respect to programming

3.1.2 Interpret and construct mathematical formulas

3.1.3 Create and implement basic algorithms

### 3.2 Search Algorithms

3.2.1 Search data in an array  
3.2.2 Describe the efficiency of linear vs. binary searches [e.g., O(n) and O(log n)]

3.2.3 Compare and contrast different algorithms for searching data

3.2.4 Analyze the efficiency of search algorithms

### 3.3 Sort Algorithms

3.3.1 Sort data in an array

3.3.2 Describe the efficiency of different sorting algorithms (e.g., bubble, insertion, and merge)

3.3.3 Compare and contrast different algorithms for sorting data

3.3.4 Analyze the efficiency of sort algorithms

# Data and Analysis

Synthesize concepts, practices and processes of data collection, resource management, and techniques to different types of data in order to discover useful information that can communicate storytelling and to inform decision-making.

## 4.0 Data and Analysis

### 4.1 Storage

4.1.1 Save, retrieve, copy, and delete files from a computing device.

4.1.2 Explain how computers store information in bits and bytes and define what information is stored.

4.1.3 Describe how numbers, text, and media are represented in bits/bytes and stored as files.

4.1.4 Assess the benefits and drawbacks of various storage models, including cloud storage, by considering factors such as cost, speed, reliability, accessibility, privacy, and integrity.

### 4.2 Collection, Visualization, And Transformation

4.2.1 Collect data using an appropriate tool and organize it.

4.2.2 Collect, organize, and present data in at least three different formats and use it to support a claim or tell a story.

4.2.3 Analyze how data collection and visualization/storytelling can be shaped by human motive, perspective, and bias.

4.2.4 Develop a simple algorithm or program that allows them to organize and represent a dataset to analyze findings, predict future outcomes, or infer trends

4.2.5 Describe the attributes that define big data, including volume, velocity, variety, veracity, and value and consider how big data has transformed our everyday lives.

### 4.3 Inference And Models

4.3.1 Identify patterns in data, charts, and/or graphs.

4.3.2 Create a simple model that organizes patterns observed in data, charts, and/or graphs.

4.3.3 Make predictions based on patterns found in data, charts, and/or graphs.

4.3.4 Create a computational model based on patterns observed in data and use it to predict future outcomes.

4.3.5 Create a computational model based on patterns observed in data and use it to predict future outcomes.

### 4.4 Data Validity

4.4.1 Identify ways their digital or physical activity creates digital data and how to adjust privacy settings on commonly used digital tools.

4.4.2 Discover who owns the digital data they produce and how it is used. Identify whether that data is open or anonymized to remove their personal information.

4.4.3 Assess data validity laws and regulations.

4.4.4 Understand and be able to advocate for their data rights and the rights of others.

### 4.5 Applications of AI

4.5.1 Identify commonly used digital tools that incorporate AI (artificial intelligence) and machine learning and acknowledge how such tools help people accomplish tasks that only humans could previously do.

4.5.2 Define AI and give real-world examples of how it has been used to extract information from data.

4.5.3 Describe basic algorithms of AI systems and how data and machine learning interact.

4.5.4 Assess how human biases are embedded within technical systems and artificial intelligence.

# Networks and the Internet

Apply networking concepts, using various models to implement protocols and standards when moving data. Design systems with working switching and routing "packets" to ensure data flows to the correct destination. Ensure data traffic flows through the internet effectively.

# Computing Systems

Apply concepts of physical components and software that make up a computing system which communicate and process information in digital form, along with practices and methodology for troubleshooting issues in those systems.

# Cybersecurity

Prove how to detect, prevent and mitigate threats in order to secure a computing system or network in an ethical manner, and in accordance with international, federal,state, local and cyber laws and regulations.

## 5.0 System and Security

### 5.1 Hardware and Software

5.1.1 Explain the composition of a computer system including physical and nonphysical components.

5.1.2 Describe and model how hardware and software work together (i.e., sending, receiving, processing, storing information as bits)

5.1.3 Describe and model how application software, systems software, and hardware interact together

5.1.4 Program a physical device that collects and exchanges information between its hardware and software components.

5.1.5 Create an integrated and embedded system that consists of multiple physical devices that collect and exchange information.

### 5.2 Connected Devices

5.2.1 Use a computing device to complete an everyday task and compare different types of devices and explain how they might meet or fail to meet certain users’ needs or preferences.

5.2.2 Connect a device to another device through a physical or wireless connection or to a cloud-based platform to extend the device’s capabilities.

5.2.3 Create a system by connecting multiple computing devices through physical or wireless connections.

5.2.4 Model how connected devices collect and exchange information with each other without human interaction (commonly known as the Internet of Things (IoT)).

5.2.5 Explore the benefits and drawbacks of working with connected devices that exchange information with each other without human interaction.

### 5.3 Troubleshooting

5.3.1 Identify and describe common problems associated with computers, external devices, and networks.

5.3.2 Use common troubleshooting strategies to solve simple problems with computers, external devices, and networks.

5.3.3 Research and implement solutions to simple problems with computers, external devices, and networks.

5.3.4 Document the steps followed to solve a problem in a way that allows others to solve a similar technical problem.

5.3.5 Research and implement solutions to complex problems with computers, external devices, and networks.

### 5.4 Digital Connectivity

5.4.1 Use a digital tool to interact with others and describe how digital tools have changed how people communicate with each other.

5.4.2 Trace the history and evolution of social media and assess how it has shaped how people communicate and share information

5.4.3 Explore how digital tools can be used to promote inclusive and exclusionary communities. Identify examples of inclusive online messaging as well as examples of cyberbullying and/or hateful speech.

5.4.4 Use a digital tool to collaborate with others and build a simple digital project or complete a class project.

### 5.5 Cybersecurity

5.5.1 Define cybersecurity and create safe passwords using effective criteria.

5.5.2 Describe common types of cyberattacks and identify malicious content (e.g., spam, spyware, viruses, phishing, etc.).

5.5.3 Apply common prevention practices (i.e.,: antivirus software and encryption) that prevent or minimize the impact of cyberattacks.

5.5.4 Assess the role that people play in creating, preventing, and minimizing the impacts of cyberattacks as well as consider how they affect people and society.

5.5.5 Define major cybersecurity risks and recommend security measures that can be taken to prevent them.

# Programming Curriculum Framework

## Program of Study

The program of study illustrates the sequence of academic and career and technical education coursework that is necessary for the student to successfully transition into postsecondary educational opportunities and employment in their chosen career path.

## Program Structure

The core course sequencing provided in the following table serves as a guide to schools for their programs of study. Each course is listed in the order in which it should be taught and has a designated level. Complete program sequences are essential for the successful delivery of all state standards in each program area.

**Programming I**

**Programming II**

**Programming III (optional)**

The core course sequencing with the complementary courses provided in the following table serves as a guide to schools for their programs of study. Each course is listed in the order in which it should be taught and has a designated level. A program does not have to utilize all of the complementary courses in order for their students to complete their program of study. Complete program sequences are essential for the successful delivery of all state standards in each program area.

## Programming I

This course will introduce students to the essential ideas of Programming and show how computing and technology can influence the world. This course focuses on technology and programming as a means to solve computational problems and find creative solutions. The appropriate use of technology and industry-standard equipment is an integral part of this course.

### Technical Competencies

**1.0 Program Design**

**2.0 Programming**

**3.0 Algorithms**

**4.0 Data and Analysis**

**5.0 Systems and Security**

### CTE Professionalism and IT Essentials Competencies

Terminology and Communications

Tools and Equipment

Project Management

Applied Mathematics

Safety

## Programming II

This course follows The College Board Advanced Placement curriculum and prepares students for the AP Programming Principles exam. This course will introduce students to the essential ideas of Programming and show how computing and technology can influence the world. This course focuses on technology and programming as a means to solve computational problems and find creative solutions. Students will creatively address real-world issues and concerns while using the same processes and tools as artists, writers, computer scientists, and engineers to bring ideas to life. The appropriate use of technology and industry standard equipment is an integral part of this course.

### Technical Competencies

**1.0 Program Design**

**2.0 Programming**

**3.0 Algorithms**

**4.0 Data and Analysis**

**5.0 Systems and Security**

### CTE Professionalism and IT Essentials Competencies

Terminology and Communications

Tools and Equipment

Project Management

Applied Mathematics

Safety

## Programming III

This course is a continuation of Programming I or AP Programming Principles. This course provides intermediate Programming students with instruction in advanced techniques and processes, particularly as it relates to the language of Java. The areas of major emphasis in the course will be on object-oriented programming methodology, algorithms, data structures and ethics. Topics will include program design, program implementation, standard data structures, and standard algorithms. The appropriate use of technology and industry-standard equipment is an integral part of this course.

### Technical Competencies

Students have achieved all program content standards and will pursue advanced study through investigation and indepth research

### CTE Professionalism and IT Essentials Competencies

Students have achieved all program content standards and will pursue advanced study through investigation and indepth research.

### Sample Topics

* Internship
* Capstone Project
* Portfolio
* Class Project Manager
* Teaching Assistant
* CTSO Leadership