



Curriculum & Instruction

SUMMIT PUBLIC SCHOOLS

Content Area: Mathematics/Computer Science
Course: Introduction to Computer Programming (H565)
Grade(s): 9-12
Semester

Course Overview

This elective course has been developed to simultaneously introduce students to computer programming and to prepare them for an introductory course in the formal JAVA language. Students will learn to program stories and games in the Alice programming environment. Students will learn the basics of programming languages with a focus on critical knowledge and skills that may be applied in future classes. Students will learn to create and manipulate object classes and to use these manipulations to develop program animations. As they progress, students will learn to program using a variety of diverse techniques, including if/else statements, loops, while and for statements, as well as techniques unique to the Alice program. Completion of this class will prepare students to enroll in a follow-up course that will introduce them to the Java programming language.

Prerequisite: None

Graduation Requirement: 21st Century Life and Careers

Subsequent Course: Intro to Java Programming

New Jersey State Learning Standards

Summit Public Schools curriculum aligns with the [New Jersey Student Learning Standards](#).

Interdisciplinary Standards:

- RI.MF.9–10.6. Analyze, integrate, and evaluate multiple interpretations (e.g., charts, graphs, diagrams, videos) of a single text or text/s presented in different formats (visually, quantitatively) as well as in words in order to address a question or solve a problem.
- RST.9-10.7: Translate quantitative or technical information expressed in words in a text into visual form (e.g., a table or chart) and translate information expressed visually or mathematically (e.g., in an equation) into words.

Career Readiness, Life Literacies and Key Skills Standards:

- 9.4.12.CI.1: Demonstrate the ability to reflect, analyze, and use creative skills and ideas (e.g., 1.1.12prof.CR3a).
- 9.4.12.CT.1: Identify problem-solving strategies used in the development of an innovative product or practice (e.g., 1.1.12acc.C1b, 2.2.12.PF.3).
- 9.4.12.CT.2: Explain the potential benefits of collaborating to enhance critical thinking and problem solving (e.g., 1.3E.12profCR3.a).

Computer Science and Design Thinking Standards:

- 8.1.12.AP.1: Design algorithms to solve computational problems using a combination of original and existing algorithms.
- 8.1.12.AP.3: Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
- 8.1.12.AP.4: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by combining code snippets or remedies.
- 8.1.12.AP.5: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
- 8.1.12.AP.6: Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
- 8.1.12.CS.1: Describe ways in which integrated systems hide underlying implementation details to simplify user experiences.
- 8.1.12.CS.4: Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
- 8.1.12.DA.2: Describe the trade-offs in how and where data is organized and stored.

Career Readiness Practices

- CRP 1. Act as a responsible and contributing citizen and employee.
- CRP 2. Apply appropriate academic and technical skills.
- CRP 3. Attend to personal health and financial well-being.
- CRP 4. Communicate clearly and effectively and with reason.
- CRP 5. Consider the environmental, social and economic impacts of decisions.
- CRP 6. Demonstrate creativity and innovation.
- CRP 7. Employ valid and reliable research strategies.
- CRP 8. Utilize critical thinking to make sense of problems and persevere in solving them.
- CRP 9. Model integrity, ethical leadership and effective management.
- CRP 10. Plan education and career paths aligned to personal goals.
- CRP 11. Use technology to enhance productivity.
- CRP 12. Work productively in teams while using cultural global competence.

New Jersey Curriculum Mandates:

This curriculum integrates lessons that address the following interdisciplinary New Jersey Curriculum Mandates:

- [Diversity and Inclusion](#) (Grades K - 12)
 - C.18A:35-4.36a: Curriculum to include instruction on diversity and inclusion.
 - (3) encourage safe, welcoming, and inclusive environments for all students regardless of race or ethnicity, sexual and gender identities, mental and physical disabilities, and religious beliefs.

Learning Objectives

Learning objectives are clear, specific statements that define what students should know, understand, and be able to do by the end of a lesson or course.

- Students will learn how to break down algorithmic programming tasks into a sequence of instructions in pseudocode and Python code.
- Students will define and call functions to write more reusable and organized code.
- Students will write loops to reduce repetition in code.
- Students will write conditional statements to solve problems that require decision-making logic.
- Students will learn to debug code by hand-tracing code and interpreting error messages.
- Students will write output to the terminal using print statements.
- Students will store and give names to data values in their programs by creating and updating variables.
- Students will be able to identify basic data types in Python.
- Students will write input statements to read data values from the terminal.
- Students will use typecasting to convert between data types as needed to process user input, write print statements, and perform mathematical operations.
- Students will use mathematical operators to transform values from user-defined variables and inputs.
- Students will write boolean conditions to make decisions in their programs.
- Students will use comparison operators in conditions to evaluate user input and current variable values.
- Students will use logical operators to combine and/or negate conditions.
- Students will use else and elif statements to introduce multiple branching decision paths in their code.
- Students will use if-elif-else and consecutive if structures as appropriate to solve logical problems.
- Students will write for loops to repeat a code block a set number of times, based on a pre-defined or user-inputted number.
- Students will write while loops to repeat a code block as long as a specified boolean condition is met.
- Students will implement loop-and-a-half structures to repeat a code block until a specified boolean condition is met.
- Students will use break statements to terminate a loop based on a variable or user-inputted value.
- Students will write nested control structures to handle multi-step decisions and repeated patterns.
- Students will combine loops, conditional statements, and user input to create an interactive game.

- Students will write functions with parameters to create reusable commands that can accept different input values.
- Students will identify and account for the difference between the scope of local and global variables.
- Students will write functions with return values to produce an output that is used elsewhere in the program.
- Students will handle predictable errors caused by user input using try-except statements.
- Students will access individual characters and slices of a String using index values.
- Students will manipulate Strings by using slicing and concatenation.
- Students will use for and for-each loops to iterate through the characters in a String.
- Students will call built-in String methods to write more efficient code and implement more complex algorithms.
- Students will create lists to store related data values.
- Students will use built-in Python list methods to add and remove values from lists.
- Students will traverse lists using for and for-each loops to summarize and transform list data.

Units of Study

Units of study are organized collections of lessons and learning experiences focused on a specific topic and set of skills within a course. Each unit is guided by a set of essential questions.

Unit 1: Introduction to Programming (~12 Days)

- What is computer programming?
- How can we define and break down a problem?
- How can we reduce repetition in our code?
- How can we add decision-making logic into our code?
- What techniques can be used to deal with errors in our code?

Unit 2: Basic Python and Console Interaction (~6 Days)

- How can we write interactive programs with the Python terminal?
- How can we use variables to store pre-defined values and inputs from the terminal?
- What are the basic data types in Python?
- When and why do we need to convert between data types?
- How can we perform mathematical operations in Python?

Unit 3: Conditional Statements (~11 Days)

- How can we make programs behave differently depending on a specified condition?
- How can we check the values of inputs and variables to influence program behavior?
- How can we account for logical scenarios that require us to check multiple aspects about the state of the program?

Unit 4: Loops (~9 Days)

- How can we use loops to reduce repetition in our code?
- How can we write and use conditions to control when a loop continues or terminates?
- How can we exit the execution of a loop early when necessary?

Date Revised: September 2026

- How can we combine conditional statements and loops to create more interactive and useful programs?

Unit 5: Functions and Exceptions (~6 Days)

- How can we use functions to make our code more modular, flexible, and reusable?
- What is the difference between defining a variable inside and outside of a function?
- How can we handle predictable errors caused by user input?

Unit 6: Strings (~11 Days)

- How can we access individual characters and use slices of a String?
- How can we analyze and transform Strings using traversals?
- How can we use built-in String methods to write more efficient code?

Unit 7: Lists (~9 Days)

- How can we use lists to store sets of data in our program?
- How can we access and manipulate the values stored in lists?
- How can we use built-in list methods to write more efficient code?

Instructional Resources and Materials

Instructional resources are the information sources teachers utilize to guide their teaching and assessing. Instructional materials are the texts students read or review. The instructional resources and materials may vary in order to meet the learning needs of students.

- Guided Note Outlines
- Google Classroom
- CodeHS
- Khan Academy

Assessment Strategies

Assessment measures a student's mastery of a course standard and learning objective. Assessment can be used for both instructional purposes and for evaluative purposes. Teachers may choose one or more of the following ideas to assess learners. They may also modify these ideas to make the assessment more responsive to students' interests and needs.

Formative:

- Practice Problem Sets,
- Homework Exercises,
- Exit Tickets,
- Graded Check-Ins

Summative:

- Quizzes
- Unit Tests

Date Revised: September 2026

Benchmark:

- Final Exam

Alternative:

- Projects

Suggested Strategies for Modification & Differentiation

[Modifications Chart](#)