SCOPE & SEQUENCE – 3-8

Coding Foundations

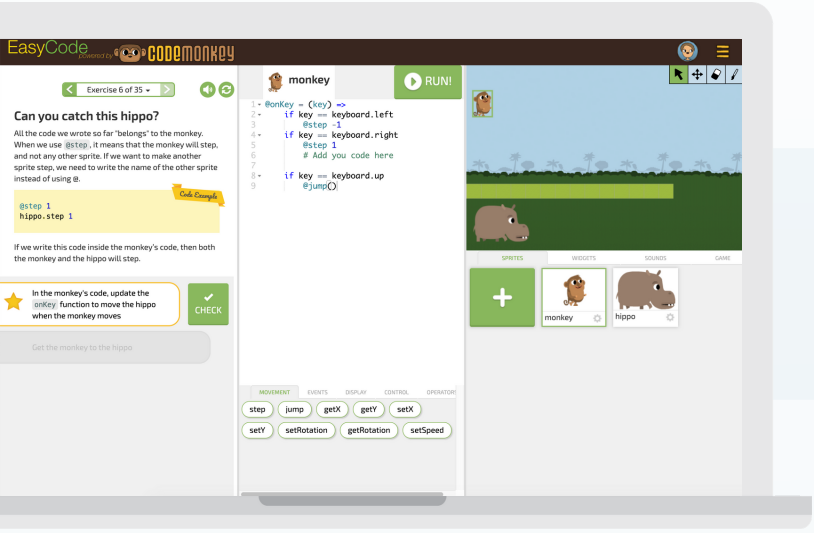
Learning.com's EasyCode Foundations for grades 3-8 builds career-ready coding skills in a text-based coding environment. Whether it is a students' first introduction to coding or their transition from block-based coding, EasyCode Foundations builds students' skills in an engaging game-based environment.

EasyCode Foundations curriculum complements the problem-solving and algorithmic thinking practice in EasyCode's Computational Thinking content area.

Through the sequence, students become novice developers equipped to design virtual worlds, program real solutions, and – of course – graduate future ready.

In conjunction with CodeMonkey, Learning.com's instructional content includes:

- **Game-Based Challenges:** Self-paced and interactive coding challenges that introduce students to programming concepts
- **Challenge Builder:** Collaborative exercises that encourage students to practice design thinking and seek feedback from peers
- **Platformer Course:** Guided projects that offer students the opportunity to leverage their new knowledge and skills
- **Game Builder:** Creative, student-led projects that offer students the opportunity to leverage their new knowledge and skills



*For a comprehensive look at EasyCode Foundations through the Learning.com platform - including the built-in gradebook, dynamic progress monitoring, and district-wide reporting - contact us for a live demo, **sales@learning.com**.*

ABOUT LEARNING.COM

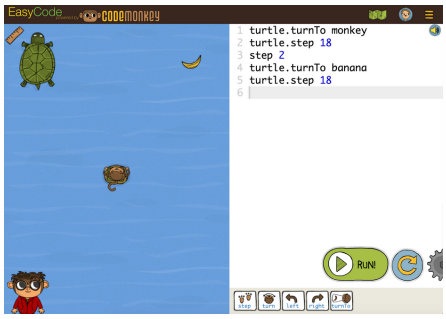
Learning.com's digital literacy curriculum enables schools to develop students' technology skills throughout core instruction.

sales@learning.com • **800.580.4640** • **learning.com**

Learning.com

UNIT 1

Coding Basics



Big Idea: Text-based coding languages communicate with computers through sequences of directions.

Skills

- Model daily processes by creating and following algorithms to complete tasks.
- Develop plans that describe a program's sequence of events, goals, and expected outcomes.

Key Topics

- Programming
- Functions
- Arguments of Functions
- Planning
- Sequences
- Dot Notation

Lessons

- Let's Get Started: Challenges 0 – 5
- Turn Around: Challenges 6 – 10
- I Have a Plan: Challenges 11 – 15
- Turtle Lake: Challenges 16 – 20

UNIT 2

Loops

Big Idea: Finding patterns in sequences enables programmers to simplify code with repetition.

Skills

- Develop programs with sequences and simple loops, to express ideas or address a problem.
- Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.

Key Topics

- Loops

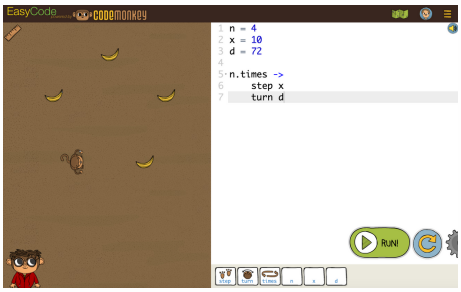
Lessons

- In the Loop: Challenges 21 - 25
- Loop on: Challenges 26 - 30

UNIT 3

Variables

Big Idea: Variables are abstractions used to represent data and information.



Skills

- Model the way programs store and manipulate data by using numbers or other symbols to represent information.
- Create programs that use variables to store and modify data.

Key Topics

- Variables
- Return Values

Lessons

- Variable Valley: Challenges 31 - 35
- Drop It: Challenges 36 - 40
- Walk the distanceTo: Challenges 41 - 45

UNIT 4

Arrays

Big Idea: Arrays are representations of categorized data types.

Skills

Model the way programs store and manipulate data by using numbers or other symbols to represent information.

Key Topics

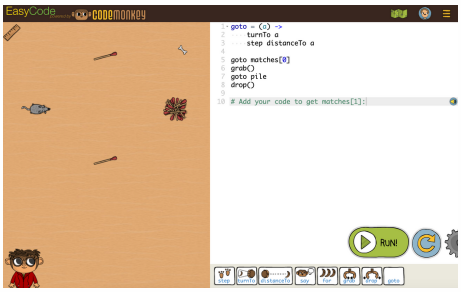
- Arrays
- For Loops
- Array Elements
- Array Indexing
- Nested Loops

Lessons

- Change Is All Around: Challenges 46 – 50
- A for Array: Challenges 51 – 60
- For Loop to the Rescure: Challenges 61-65
- Iterate Mate: Challenges 66 – 70
- Crocodile Rock: Challenges 71 – 75

UNIT 5

Functions



Big Idea: Functions are decomposed steps in algorithms that represent modules performing a certain task.

Skills

- Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
- Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.

Key Topics

- Defining Functions
- Commenting
- Debugging
- Calling Functions

Lessons

- Let's Build Something: Workshop I
- Function Farm: Challenges 76-80
- Fun-ctions: Challenges 81-85
- Fake It 'Til You Make It: Challenges 86-90

UNIT 6

Loops

Big Idea: Arrays are representations of categorized data types.

Skills

Model the way programs store and manipulate data by using numbers or other symbols to represent information.

Key Topics

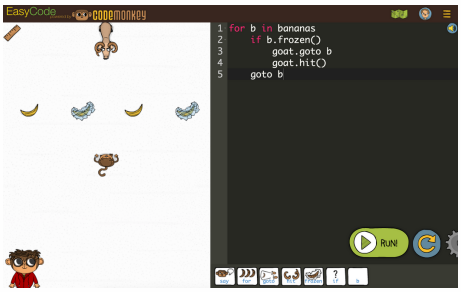
- Arrays
- For Loops
- Array Elements
- Array Indexing
- Nested Loops

Lessons

- It Ain't Over Until It's Over: Challenges 91-95
- Cut to the Chase: Challenges 96 – 100
- Wait() for It: Challenges 101 – 105

UNIT 7

If Statements



Big Idea: If statements utilize conditional logic to guide the functions in a program.

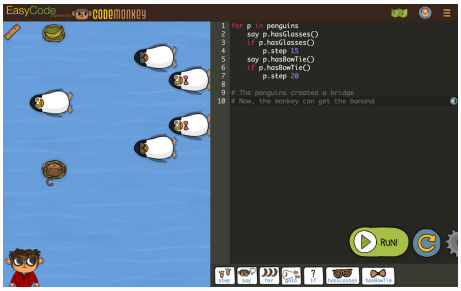
Skills
Create programs that include sequences, events, loops, and conditionals.

Key Topics
- If Statements
- If-Else
- For loops

Lessons
- Act the Goat: Challenges 106 – 115
- Green Banana Sorbet: Challenges 116 – 121
- If All Else Fails: Challenges 122 – 124

UNIT 8

Logic



Big Idea: Logic draws facts or rules about given statements.

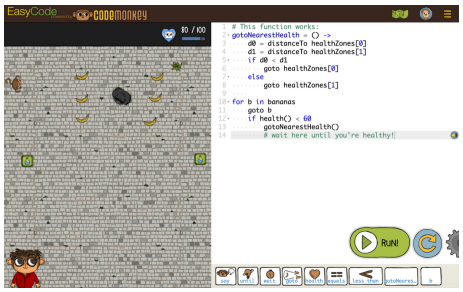
Skills
Create programs that include sequences, events, loops, and conditionals.

Key Topics
- Logic Operators: until, and, or
- Boolean Logic
- Polymorphism

Lessons
- And in the End: Challenges 125 – 130
- Take it or Leave it: Challenges 131 – 135
- Mix and Match: Challenges 136-140
- Fashion Alert: Challenges 141 – 145

UNIT 9

Logical Operators



Big Idea: Logic draws facts or rules about given statements.

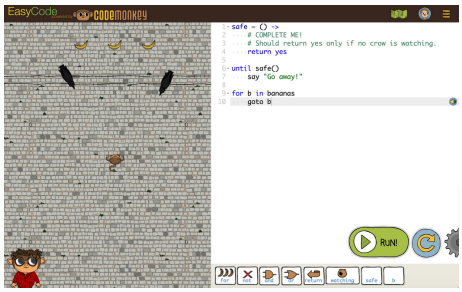
Skills
Create programs that include sequences, events, loops, and conditionals.

Key Topics
- Not statement
- Comparator (==)
- Less than (<)

Lessons
- Not Bad: Challenges 146 – 151
- Health() == Happiness: Challenges 152 – 156
- Less is More: Challenges 157 – 160

UNIT 10

Return Values



Big Idea: Return statements move programs from sub-functions to the larger function.

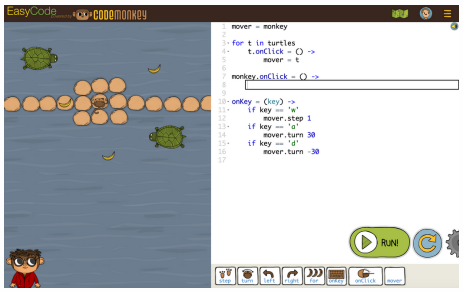
Skills
Create programs that include sequences, events, loops, and conditionals.

Key Topics
- Return Values
- Calling Functions
- Not / And

Lessons
- Return to Sender: Challenges 161 – 165
- Scarecrow Monkey: Challenges 166 – 170
- Scare Them All: Challenges 171 - 175

UNIT 11

Keyboard Events



Big Idea: Actions taken by a user trigger functions in a program.

Skills

- Create programs that include sequences, events, loops, and conditionals.

Key Topics

- Keyboard Events
- Mouse-Click Events

Lessons

- Im-Pressing: Challenges 176-180
- You Can Control Me: Challenges 181-184
- Behind Gate #1: Challenges 185-189
- Monkey, It's Time to Rest: Challenges 190-195
- The Bananas Are Just a Click Away: Challenges 196-200
- Who Moved My Monkey?: Challenges 201-205

MASTERY

Review Opportunities

At intervals in EasyCode Foundations, students revisit past lessons to refresh and work toward mastery of the concepts.

Lessons

- Wait() For it: Challenges 101-105
- Fundamentals Stars Party: Challenges 1-75 and Skill Challenges
- Functions & Conditionals Stars Party: Challenges 76-145 & Skill Challenges
- Logic & Events Stars Party: Challenges 146-210 and Skill Challenges
- Gorilla, We Are Ready for You: Challenges 206-210

SUMMATIVE

Challenge Builder

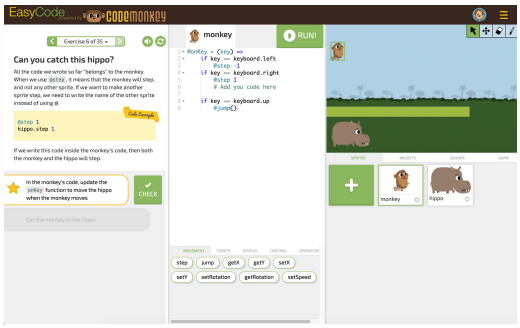
Using EasyCode's Challenge Builder, students apply their knowledge and skills to create original works of their own.

Projects

- Let's Build Something : Workshop I
- Loops Are Fun: Workshop II
- What's the Conditional: Workshop III
- Escape the Maze: Workshop IV
- Debate - Keyboard or Mouse: Workshop V
- Game Away: Workshop VI

Game Builder

Students collaborate or work individually to build games they and their peers can play. Students create in the Game Builder, they collaborate with peers throughout the iterative development process, they communicate their decisions and reasons, they think critically for developing program sequences, and – of course – they think computationally.



Courses

- Platformer Course
- Frogger Course
- Draw Your Own Sprites Course

Communication

• Using correct terminology, describe steps taken and choices made during the iterative process of program development.
• Document programs in order to make them easier to follow, test, and debug.
• Describe choices made during program development using code comments, presentations, and demonstrations.

Critical Thinking

• Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
• Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
• Systematically test and refine programs using a range of test cases.

Collaboration

• Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.
• Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development.
• Seek and incorporate feedback from team members and users to refine a solution that meets user needs.

Creativity

• Create clearly named variables that represent different data types and perform operations on their values.
• Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
• Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences.