

ICT Programming & Logics Essentials

Lesson 1: Introduction to Programming

LESSON SKILLS

After completing this lesson, you will be able to:

- Define “programming” and discuss its role in computing.
- Understand the binary representation of data and programs in computers.
- Compare and contrast programming languages that are compiled and interpreted.
- Describe the structure of a simple program, and understand why sequencing is important.
- Describe structured programming and discuss the advantages of this approach.
- Write a program in pseudocode that uses structured programming to solve a problem.

KEY TERMS

assembly language	execute	sequential
binary	high-level language	structured programming
bits	instructions	syntax
bytes	interpreter	transistor
compiler	machine language	translate
data	programming language	

Points to Ponder

These Points to Ponder are designed to help you focus on key elements in this lesson. They are also suitable for use to spark discussions or individual research.

- Define “programming” in your own words and give examples of its role in daily computing.
- Explain how the binary number system differs from the decimal number system. Show the process to convert the number 12 to binary.
- Describe the difference between a compiler and an interpreter.
- Which type of programming language is the only language that computers understand, and why?
- Explain why the binary number system is used by computers.
- List and describe the three basic types of programming languages.
- Create a diagram that provides instructions for traveling to the school's main office from your classroom.
- Explain the binary number system. How are bits and bytes used by the computer?

Overview

This lesson introduces you to computer programming and the basics of programming languages. You will also learn how to deconstruct an existing program so they can see how it works, and thus begin learning to construct programs to accomplish specific tasks.

What Is Programming?

Objectives

4.1.1: Define "programming," and discuss its role in computing.

This section lays the foundation for the remaining lessons by defining programming and the syntax used to write programs. You will also explore the role of programming in everyday computing, and you will deconstruct a simple program.

Computer programs are in use around us daily. There are many examples of how computer programs are used every day. When thinking about programming, many people focus on Personal Computers, but programming is used to control devices from refrigerators to home security. Programmed robots are used to manufacture cars ([How Cars Are Built](#), YouTube video, 6 mins); and allow cars to drive themselves ([Google Self-Driving Car](#), YouTube video, 3 mins).

Software and applications, such as those used in the examples mentioned above, consist of computer programs that provide instructions to the computer. A computer program is a set of instructions that tells a computer what to do to accomplish a particular task or solve a particular problem (Webopedia.com, n.d.). Programming is an important part of Computer Science.

Links to Learn More

Watch [What Most Schools Don't Teach](#) (YouTube video, 6 mins). What are your views about the video and your feelings about learning to program?

Visit [Blockly](#) and solve two or three maze demos to see how learning to program can be easy and fun. Before running the code for Level 2, click the Show JavaScript Code button (the left button indicated by the down-pointing arrow in Figure 1-1) to see the programming code written behind the scenes.



Figure 1-1: Blockly Show Code button

A computer program's instructions are written using a programming language. Programming languages are a combination of vocabulary and grammatical rules specific to computer instruction (Webopedia.com, n.d.). Most programming languages are written using text that looks like a lock combination or cryptic code with a bit of English thrown in, as shown in Figure 1-2.

```
turnRight();  
for (var count = 0; count < 5; count++) {  
    moveForward();  
}
```

Figure 1-2: Example of program language text

Programming languages write instructions using a particular syntax, which is a set of rules that must be followed. You could compare programming syntax to the rules for writing, such as how words are placed in a sentence to convey meaning. Different languages, such as English or Japanese, require different syntax. Example: Jack loves dogs (written subject-verb-object in English); and Jack dogs loves (written subject-object-verb in Japanese).

When learning a programming language, you must learn the syntax rules for that particular language. Learning to write programs using a programming language can be challenging, just as learning a new language such as Japanese can be challenging. It requires hard work, perseverance and lots of practice. This course will use Blockly and Scratch to teach programming. After some practice, you may want to learn to program using programming languages.

Note: Continue learning using Blockly by completing the [Tutorials for Beginners](#) at Code.org. For your convenience, as programming concepts are introduced throughout this course, the lessons contained in [Code.org's K-8 Intro to Computer Science Course](#) will be referenced. Each lesson contains a video designed to introduce you to the concepts and purpose of the lesson. A lesson plan is included on the video page.

Overview of Programming Languages

Objectives

4.1.2: Explain the binary representation of data and programs in computers.

4.1.3: Distinguish among the three types of programming languages (machine, assembly, high-level), and give examples.

4.1.4: Compare and contrast languages that are usually compiled (e.g., C++, Java) and interpreted (e.g., JavaScript, Python).

Now it is time to delve a bit deeper into what happens behind the scenes. This section introduces the three main types of programming languages, and explains how the programs are translated using compilers and interpreters.

Note: These concepts are technical and the depth of coverage may need to be adjusted to suit your students. However, a basic overview of these concepts should be completed by all students.

A computer is a machine that cannot perform a task on its own. Without a computer program, the computer cannot do anything. In order for a computer to complete a task, it must follow the instructions that are given to it and those instructions must be given in a specific manner. Do not confuse this requirement with the previous discussion involving syntax, which specifies how to write instructions according to rules of a specific programming language. This discussion is about how the computer understands and performs instructions.

When a computer is performing the instructions contained in a program, we call it executing or running the program. A computer's primary function is storing, retrieving and displaying data. Computers can also perform calculations and manipulate data, such as adding, multiplying, comparing and moving data.

There are three fundamental types of programming languages: machine language, assembly language and high-level language.

Machine Language

At the most basic level, the information given to a computer is represented by electricity. Computer chips contain billions of transistors that switch between two electrical states — off and on. The electrical state of transistor switches is represented as 0 (off) or 1 (on), which creates the language of computers. The transistors continually switch between these two states, providing the information and instructions a computer needs to complete complex calculations.

A [machine language](#) program uses a binary coding system to represent the data in the transistor switches. The binary system uses zeroes (off) and ones (on), called bits, in a sequence to represent data. A bit (binary digits) represents the state of one switch, which is the smallest unit. A byte is a series of eight bits that represent the basic unit of data a computer processes. Each byte is a combination of zeroes and ones that represents an instruction to the computer, as illustrated in Figure 1-3. Visit the [Wikiversity page on processors](#) to learn more about transistors and binary code.

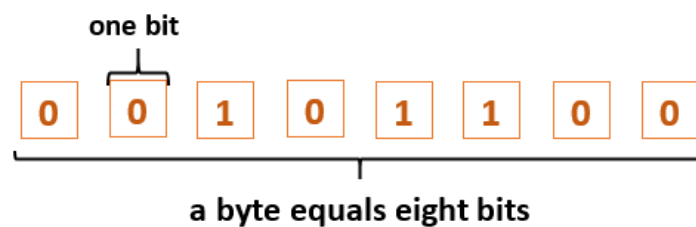


Figure 1-3: Bit and byte example

Figure 1-4 shows the text "Hello, World!" written in binary code.

```
010010000110010101101100011011000110111100100000010101110110111101  
110010011011000110010000100001
```

Figure 1-4: Hello, World! in binary

Links to Learn More

Watch [What Are Binary Numbers?](#) (YouTube video, 5 mins), which reviews the base-10 (decimal) system and introduces the base-2 (binary) system.

Watch [Reaching Out — Binary Codes](#) (YouTube video, 4 mins). The video contains a lot of useful information to understand Binary Codes.

Note that machine language is the only language that computers can understand. All other types of programming must be translated (interpreted or compiled) into machine language in order for the computer to understand its instructions. Figure 1-5 shows the text "Hello, World!" written in machine language, for comparison to the same message in binary.



Binary Game



In this activity, you will play an online game to test your knowledge of binary numbers.

1. Use your web browser to access the Binary Game (<http://www.wordfreegames.com/game/binary-game.html>).

Note: This game has music that automatically plays when you open the page. Check the volume of your speakers before you open the page. You can turn the music off if you choose.

2. Read the instructions before starting the game.
3. The game is designed to test your knowledge of binary numbers. You must determine the number value of bits, or the binary value of numbers. As the game progresses, the problems get more difficult and appear on the screen more quickly.



Programming Terms Word Search



In this activity, you will familiarize yourself with programming terms. Find the words listed below.

R S S P T X H N C D C S E L E
Y E N C R A I O L O Y H P G C
P R O G R A M M I N G B A N O
E U A T I P G P T P L U O R M
R E O N U T I A E E G O E S P
P N D T I L X X A N T T S R I
R T E D E B E U A M U S A E L
E R T R O C O L R P M N F T E
T A G I N T E R P R E T E R R
N L A I T N E U Q E S H N H O
A T A D I E T U C E X E T G A
T E T H A T L H O T K G I H E
R M C L E D T M R A N R L M A
B A R E R T T I T I R N E E I
M S N O I T C U R T S N I T D

BINARY

EXECUTE

PROGRAMMING

COMPILER

INSTRUCTIONS

SEQUENTIAL

COMPUTER


INTERPRETER

SYNTA

DATA

MACHINE LANGUAGE

Case Study: Programming with Scratch — Assignment 1

1. Using your web browser, go to the Scratch home page at <http://scratch.mit.edu/>.
2. Click the **Create** button in the top-left corner.
3. Click **All Tips** on the Help screen (at the right) to access the main Tips page.
 - If the Help screen is not visible, click the question mark  icon in the top-right corner of the Scratch editor window.
4. Click **Map of Scratch Editor** in the Step-by-Step Guides list.
5. Review the main areas of the editor, and then spend some time familiarizing yourself with the Scratch interface.
6. Return to the All Tips screen and click **Getting Started with Scratch**.
7. Click the **Start Moving** button, and complete all 13 tutorials.
8. Return to the **All Tips** screen and click **Animate Your Name**.
9. Follow the steps to animate your name.