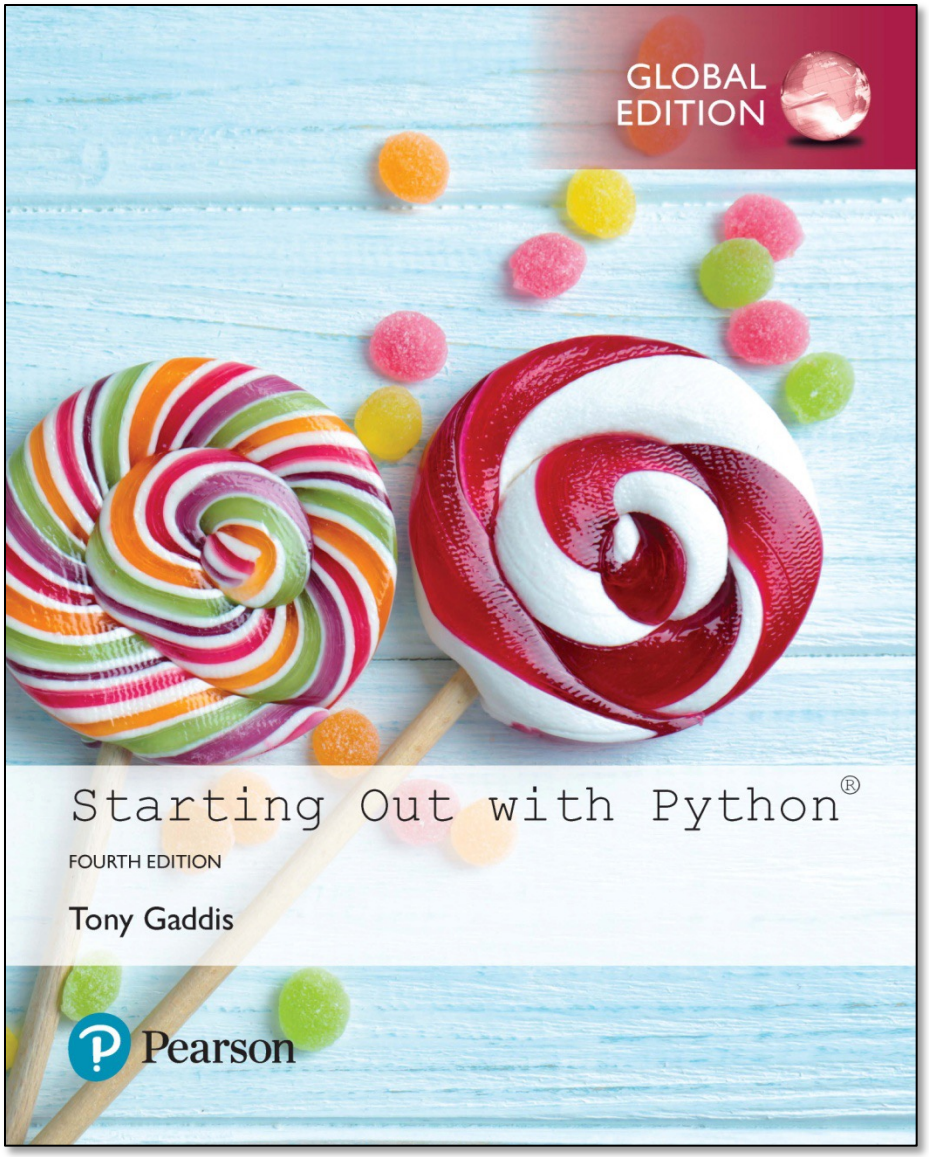
CHAPTER 1

Introduction to Computers and Programming



Topics

- Introduction
- Hardware and Software
- How Computers Store Data
- How a Program Works
- Using Python

Introduction

- **Computers can be programmed**
  - Designed to do any job that a program tells them to
- **Program: set of instructions that a computer follows to perform a task**
  - Commonly referred to as *Software*
- **Programmer: person who can design, create, and test computer programs**
  - Also known as software developer

Hardware and Software

- **Hardware: The physical devices that make up a computer**
  - Computer is a system composed of several components that all work together
- **Typical major components:**
  - Central processing unit
  - Main memory
  - Secondary storage devices
  - Input and output devices

The CPU

- **Central processing unit (CPU): the part of the computer that actually runs programs**
  - Most important component
  - Without it, cannot run software
  - Used to be a huge device
- **Microprocessors: CPUs located on small chips**

Main Memory

- **Main memory: where computer stores a program while program is running, and data used by the program**
- **Known as *Random Access Memory* or *RAM***
  - CPU is able to quickly access data in RAM
  - Volatile memory used for temporary storage while program is running
  - Contents are erased when computer is off

Secondary Storage Devices

- **Secondary storage: can hold data for long periods of time**
  - Programs normally stored here and loaded to main memory when needed
- **Types of secondary memory**
  - Disk drive: magnetically encodes data onto a spinning circular disk
  - Solid state drive: faster than disk drive, no moving parts, stores data in solid state memory
  - Flash memory: portable, no physical disk
  - Optical devices: data encoded optically

Input Devices

- **Input: data the computer collects from people and other devices**
- **Input device: component that collects the data**
  - Examples: keyboard, mouse, touchscreen, scanner, camera
  - Disk drives can be considered input devices because they load programs into the main memory

Output Devices

- **Output: data produced by the computer for other people or devices**
  - Can be text, image, audio, or bit stream
- **Output device: formats and presents output**
  - Examples: video display, printer
  - Disk drives and USB drives can be considered output devices because data is sent to them to be saved

Software

- **Everything the computer does is controlled by software**
  - General categories:
    - Application software
    - System software
- **Application software: programs that make computer useful for every day tasks**
  - Examples: word processing, email, games, and Web browsers

Software (cont'd.)

- **System software: programs that control and manage basic operations of a computer**
  - Operating system: controls operations of hardware components
  - Utility Program: performs specific task to enhance computer operation or safeguard data
  - Software development tools: used to create, modify, and test software programs

How Computers Store Data

- **All data in a computer is stored in sequences of 0s and 1s**
- **Byte: just enough memory to store letter or small number**
  - Divided into eight bits
  - Bit: electrical component that can hold positive or negative charge, like on/off switch
  - The on/off pattern of bits in a byte represents data stored in the byte

Storing Numbers

- **Bit represents two values, 0 and 1**
- **Computers use binary numbering system**
  - Position of digit `j` is assigned the value 2^{j-1}
  - To determine value of binary number sum position values of the 1s
- **Byte size limits are 0 and 255**
  - 0 = all bits off; 255 = all bits on
  - To store larger number, use several bytes

Storing Characters

- **Data stored in computer must be stored as binary number**
- **Characters are converted to numeric code, numeric code stored in memory**
  - Most important coding scheme is ASCII
    - ASCII is limited: defines codes for only 128 characters
  - Unicode coding scheme becoming standard
    - Compatible with ASCII
    - Can represent characters for other languages

Advanced Number Storage

- **To store negative numbers and real numbers, computers use binary numbering and encoding schemes**
  - Negative numbers encoded using two's complement
  - Real numbers encoded using floating-point notation

Other Types of Data

- **Digital: describes any device that stores data as binary numbers**
- **Digital images are composed of pixels**
  - To store images, each pixel is converted to a binary number representing the pixel's color
- **Digital music is composed of sections called samples**
  - To store music, each sample is converted to a binary number

How a Program Works

- **CPU designed to perform simple operations on pieces of data**
  - Examples: reading data, adding, subtracting, multiplying, and dividing numbers
  - Understands instructions written in machine language and included in its instruction set
    - Each brand of CPU has its own instruction set
- **To carry out meaningful calculation, CPU must perform many operations**

How a Program Works (cont'd.)

- **Program must be copied from secondary memory to RAM each time CPU executes it**
- **CPU executes program in cycle:**
  - Fetch: read the next instruction from memory into CPU
  - Decode: CPU decodes fetched instruction to determine which operation to perform
  - Execute: perform the operation

How a Program Works (cont'd.)

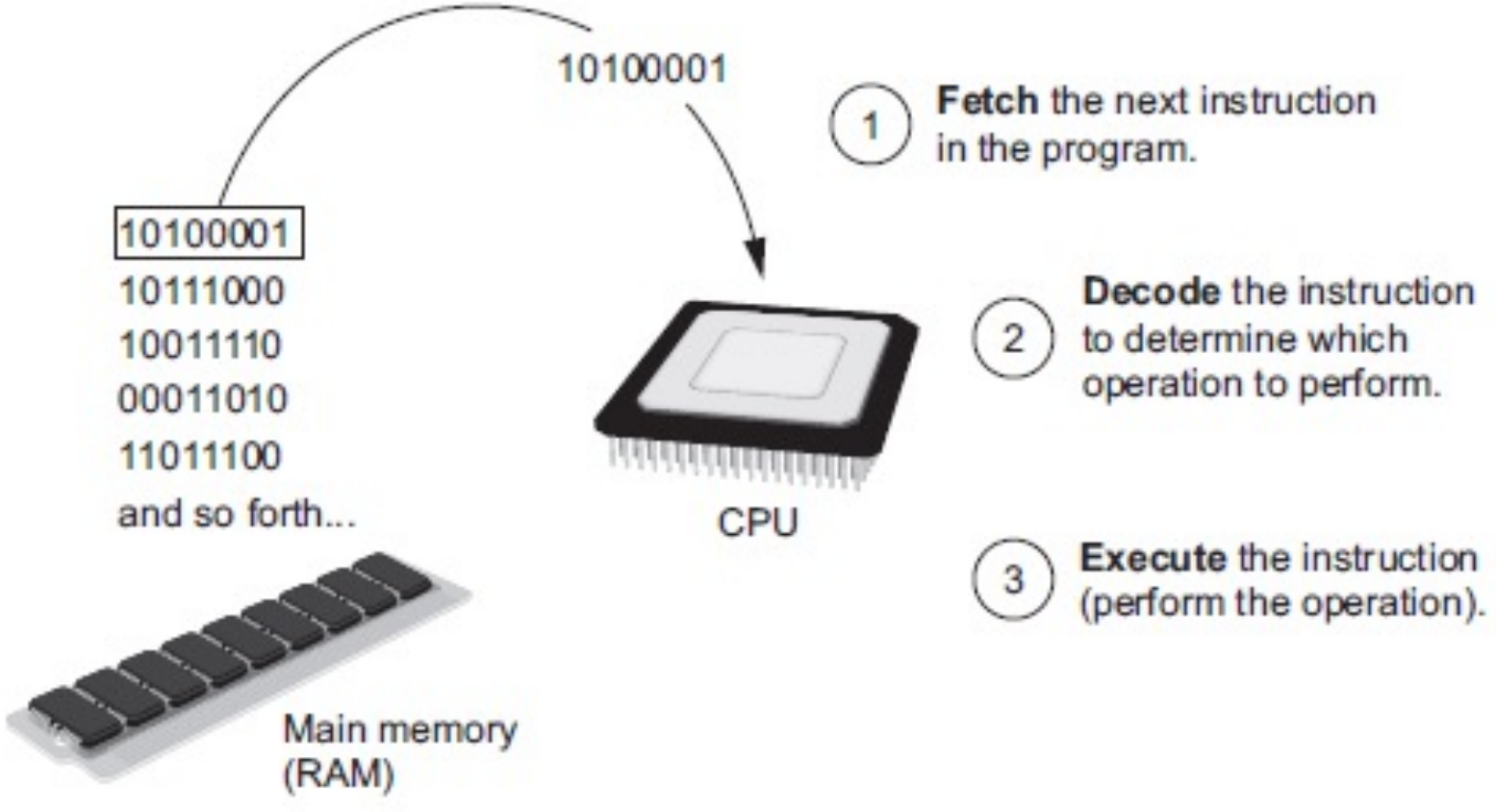


Figure 1-16 The fetch-decode-execute cycle

From Machine Language to Assembly Language

- **Impractical for people to write in machine language**
- **Assembly language: uses short words (mnemonics) for instructions instead of binary numbers**
  - Easier for programmers to work with
- **Assembler: translates assembly language to machine language for execution by CPU**

High-Level Languages

- **Low-level language: close in nature to machine language**
  - Example: assembly language
- **High-Level language: allows simple creation of powerful and complex programs**
  - No need to know how CPU works or write large number of instructions
  - More intuitive to understand

Key Words, Operators, and Syntax: an Overview

- **Key words: predefined words used to write program in high-level language**
  - Each key word has specific meaning
- **Operators: perform operations on data**
  - Example: math operators to perform arithmetic
- **Syntax: set of rules to be followed when writing program**
- **Statement: individual instruction used in high-level language**

Compilers and Interpreters

- **Programs written in high-level languages must be translated into machine language to be executed**
- **Compiler: translates high-level language program into separate machine language program**
  - Machine language program can be executed at any time

Compilers and Interpreters (cont'd.)

- **Interpreter: translates and executes instructions in high-level language program**
  - Used by Python language
  - Interprets one instruction at a time
  - No separate machine language program
- **Source code: statements written by programmer**
  - Syntax error: prevents code from being translated

Compilers and Interpreters (cont'd.)

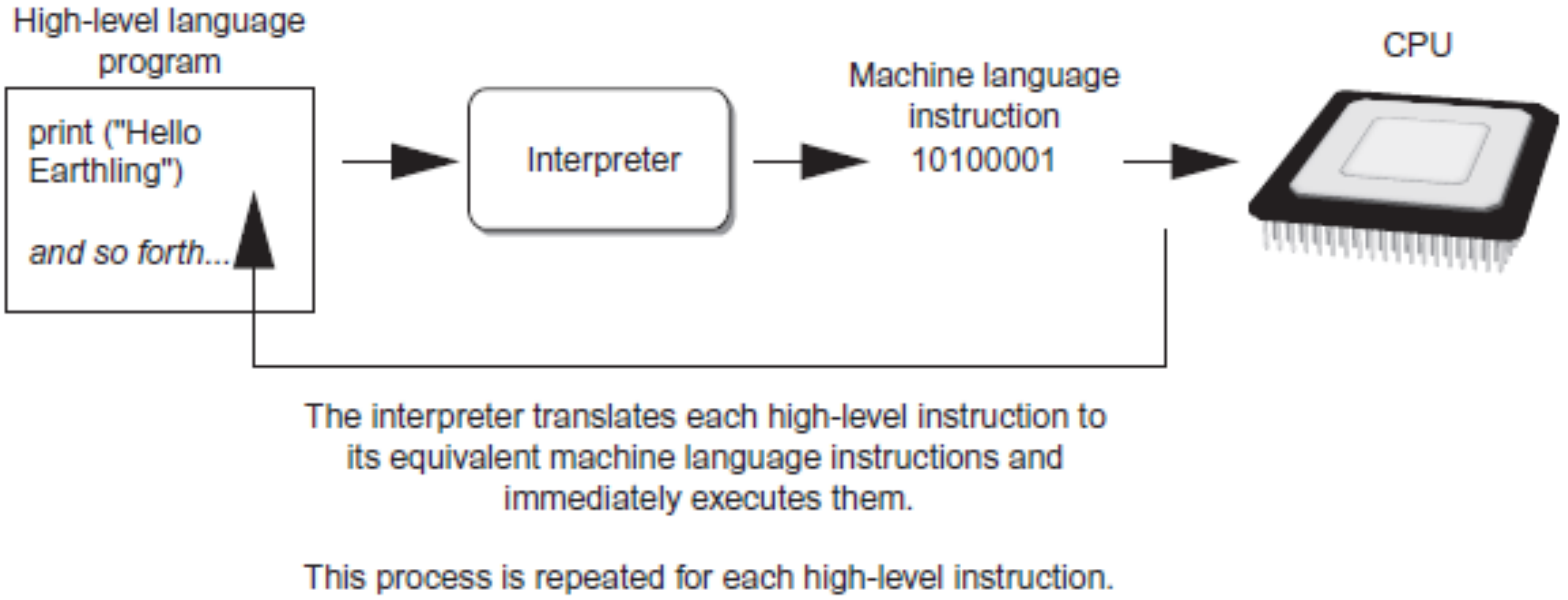


Figure 1-19 Executing a high-level program with an interpreter

Using Python

- **Python must be installed and configured prior to use**
  - One of the items installed is the Python interpreter
- **Python interpreter can be used in two modes:**
  - Interactive mode: enter statements on keyboard
  - Script mode: save statements in Python script

Interactive Mode

- **When you start Python in interactive mode, you will see a prompt**
  - Indicates the interpreter is waiting for a Python statement to be typed
  - Prompt reappears after previous statement is executed
  - Error message displayed If you incorrectly type a statement
- **Good way to learn new parts of Python**

Writing Python Programs and Running Them in Script Mode

- **Statements entered in interactive mode are not saved as a program**
- **To have a program use script mode**
  - Save a set of Python statements in a file
  - The filename should have the .py extension
  - To run the file, or script, type

    `python` *`filename`*

    at the operating system command line

The IDLE Programming Environment

- **IDLE (Integrated Development Program): single program that provides tools to write, execute and test a program**
  - Automatically installed when Python language is installed
  - Runs in interactive mode
  - Has built-in text editor with features designed to help write Python programs

Summary

- **This chapter covered:**
  - Main hardware components of the computer
  - Types of software
  - How data is stored in a computer
  - Basic CPU operations and machine language
  - Fetch-decode-execute cycle
  - Complex languages and their translation to machine code
  - Installing Python and the Python interpreter modes

Extra Slides

$0 \bmod 4 = 0$
$1 \bmod 4 = 1$
$2 \bmod 4 = 2$
$3 \bmod 4 = 3$

$4 \bmod 4 = 0$ $8 \bmod 4 = 0$
$5 \bmod 4 = 1$ $9 \bmod 4 = 1$
$6 \bmod 4 = 2$ $10 \bmod 4 = 2$
$7 \bmod 4 = 3$ $11 \bmod 4 = 3$

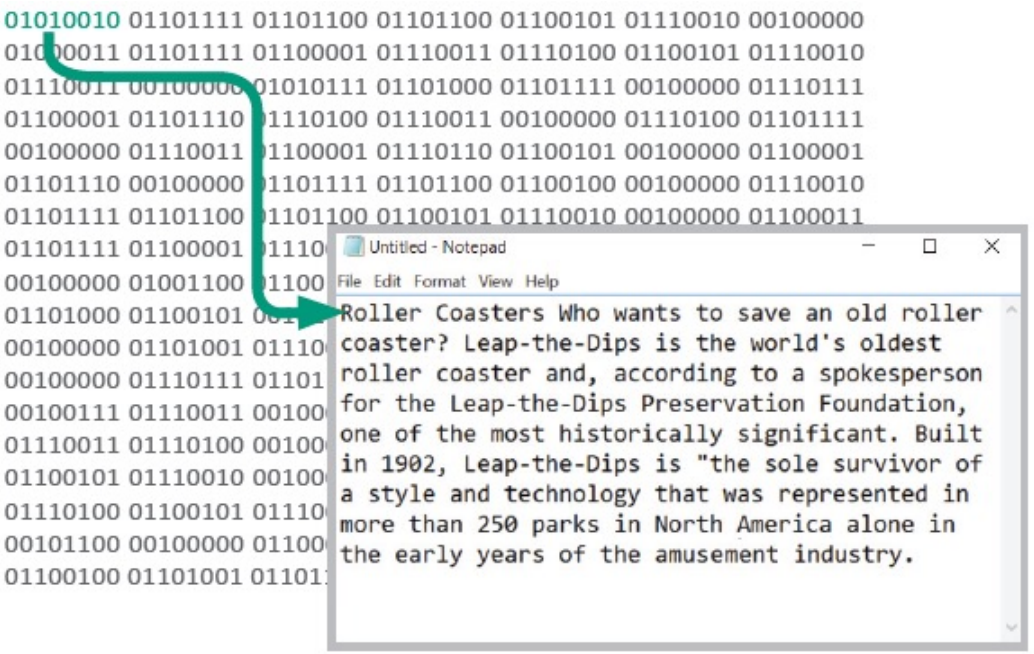
Extra Slides

DECIMAL (BASE 10)	BINARY (BASE 2)
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
1000	1111101000

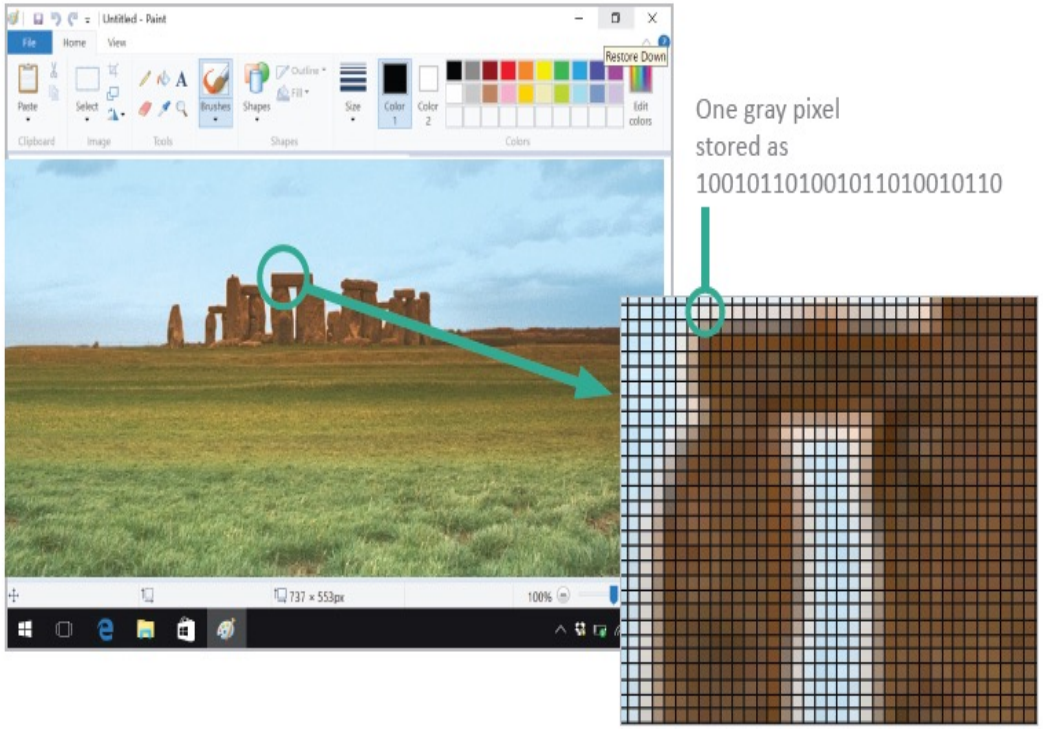
Extra Slides

00100000	Space	00110011	3	01000110	F	01011001	Y	01101100	l
00100001	!	00110100	4	01000111	G	01011010	Z	01101101	m
00100010	"	00110101	5	01001000	H	01011011	[	01101110	n
00100011	#	00110110	6	01001001	I	01011100	\	01101111	o
00100100	$	00110111	7	01001010	J	01011101	]	01110000	p
00100101	%	00111000	8	01001011	K	01011110	^	01110001	q
00100110	&	00111001	9	01001100	L	01011111	_	01110010	r
00100111	'	00111010	:	01001101	M	01100000	`	01110011	s
00101000	(	00111011	;	01001110	N	01100001	a	01110100	t
00101001	)	00111100	<	01001111	O	01100010	b	01110101	u
00101010	*	00111101	=	01010000	P	01100011	c	01110110	v
00101011	+	00111110	>	01010001	Q	01100100	d	01110111	w
00101100	,	00111111	?	01010010	R	01100101	e	01111000	x
00101101	-	01000000	@	01010011	S	01100110	f	01111001	y
00101110	.	01000001	A	01010100	T	01100111	g	01111010	z
00101111	/	01000010	B	01010101	U	01101000	h	01111011	{
00110000	0	01000011	C	01010110	V	01101001	i	01111100	|
00110001	1	01000100	D	01010111	W	01101010	j	01111101	}
00110010	2	01000101	E	01011000	X	01101011	k	01111110	~

Extra Slides



Extra Slides



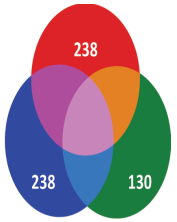
White

Red Green Blue

255 255 255

#FFFFFF

11111111 11111111 11111111



Violet

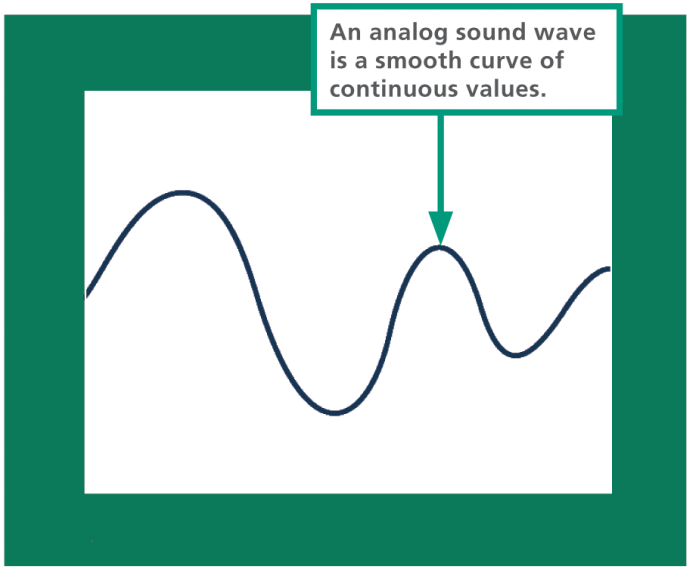
Red Green Blue

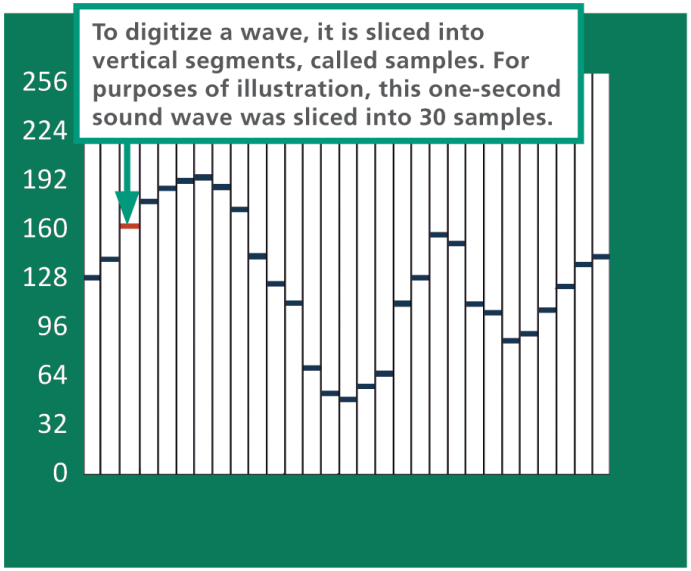
238 130 238

#EE82EE

11101110 10000010 11101110

Extra Slides





SAMPLE	SAMPLE HEIGHT (DECIMAL)	SAMPLE HEIGHT (BINARY)
1	130	10000010
2	140	1000110
3	160	10100000
4	175	10101111

The height of each sample is converted into a binary number and stored. The height of sample 3 is 160 (decimal), so it is stored as its binary equivalent—10100000.

Extra Slides

Bit	One binary digit
Byte	8 bits
Kilobit	1,024 or 2^{10} bits
Kilobyte	1,024 or 2^{10} bytes
Megabit	1,048,576 or 2^{20} bits
Megabyte	1,048,576 or 2^{20} bytes
Gigabit	2^{30} bits
Gigabyte	2^{30} bytes
Terabyte	2^{40} bytes
Petabyte	2^{50} bytes
Exabyte	2^{60} bytes

Reference

- Extra slides are taken from Introduction to MIS and Data Processing, Cengage 2019.