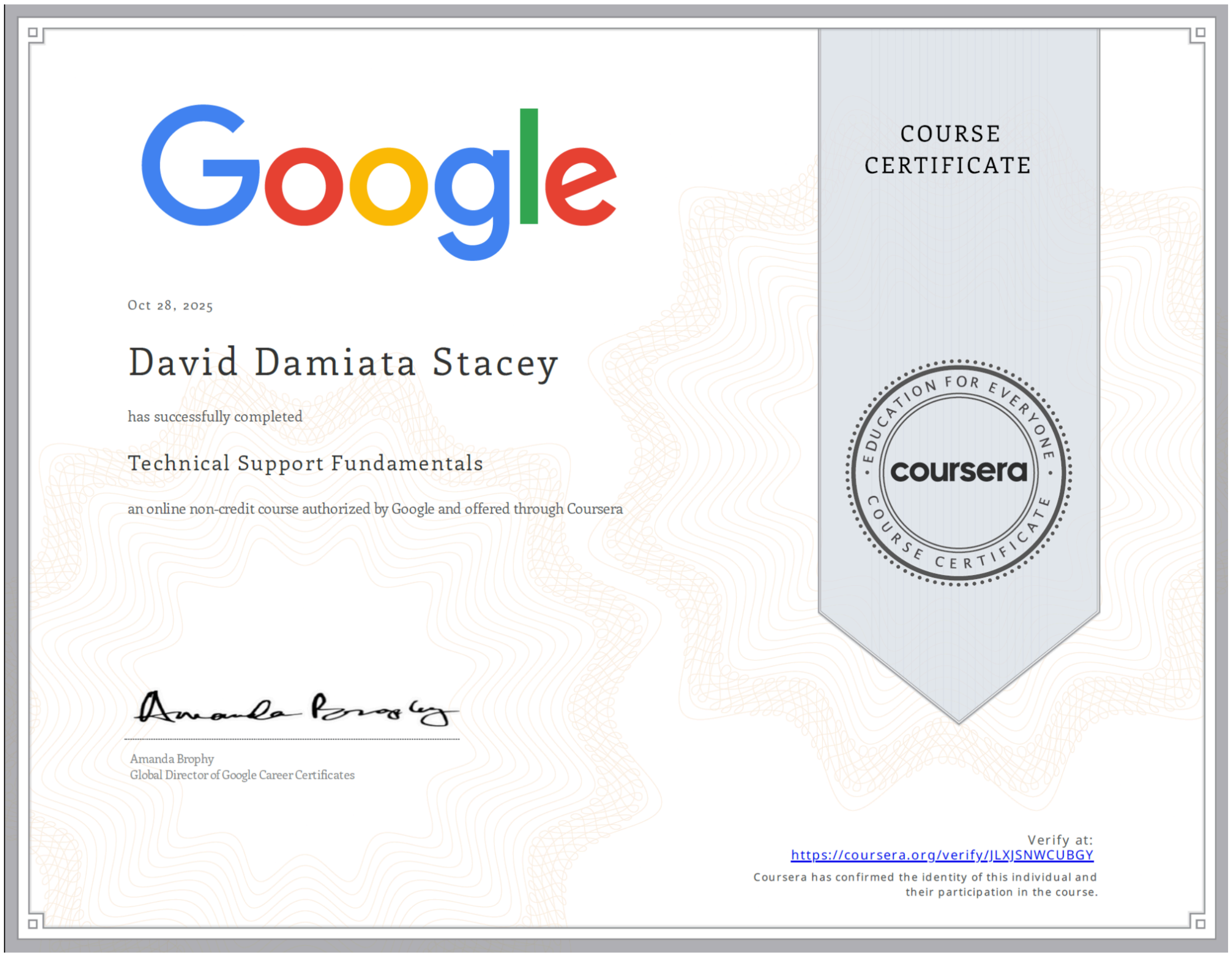
Technical Support Fundamentals

Course 1 of the Google IT Support Professional Certificate

Completed 10.28.2025



Google

COURSE CERTIFICATE

Oct 28, 2025

David Damiata Stacey

has successfully completed

Technical Support Fundamentals

an online non-credit course authorized by Google and offered through Coursera

EDUCATION FOR EVERYONE · coursera · COURSE CERTIFICATE

[illegible]

Amanda Brophy
Global Director of Google Career Certificates

Verify at:
https://coursera.org/verify/JLXJSNWCUBGY
Coursera has confirmed the identity of this individual and their participation in the course.

Pioneers in Computing and IT

Computer technology has come a long way since the first computer was invented. Along the way, many people from diverse backgrounds contributed inventions and innovations that helped us get to where we are today with modern computers. Without these individuals, information technology would not be where it is today.

Early Computer Pioneers

Ada Lovelace

Ada Lovelace was born in 1815 to Anna Milbanke and the poet Lord Byron. Her mother Anna Milbanke educated her to excel in mathematics. When Lovelace was still young, she was shown the Difference Engine (a mechanical calculator developed by Charles Babbage) and published a set of notes which contained the first computer algorithm for the Analytical Engine in 1843. Lovelace predicted at the time that computers would eventually be used outside of mathematics for things like composing music and made predictions about how technology would influence society.

Alan Turing

Alan Turing was born in 1912. While completing his degrees, he developed the concept of the Turing machine. Turing proved that there were some yes/no mathematical questions that could never be solved computationally which defined computation and its limitations. These findings would go on to become one of the seeds of computer science and his conceptual Turing machine (so named by his Doctoral advisor) is considered a predecessor of modern computer programs. During the Second World War, Turing developed the Turing-Welchman Bombe which was used to decipher Nazi codes and intercept Nazi messages. After the war, Turing's Imitation Game (now known as the Turing test) was created as a means to evaluate the abilities of artificial intelligence.

Margaret Hamilton

Margaret Hamilton was born in 1936. While working in the meteorology department at the Massachusetts Institute of Technology, she developed software for predicting weather. Later Hamilton would go on to work on the software that was used in the NASA Apollo command and lunar modules. With her experience writing software, she wanted to ensure that this skill would get its due respect and coined the term "software engineering." Culminating her experience working on the Apollo missions and moon landings, Hamilton formalized what she learned into a theory that would later become the Universal System Language.

Admiral Grace Hopper

Grace Hopper was born in 1906. During the Second World War, she joined the US Navy Reserve after taking a leave from her role as a mathematics professor at Vassar College. In the Navy, she was assigned the Bureau of Ships Computation Project at Harvard University where she worked on the programming team for the Mark I computer. After the war and her time at Harvard, she began working on more powerful computers and recommended that a programming language be developed that used English words rather than symbols. This concept would eventually become FLOW-MATIC the first programming language to use English words which also necessitated the invention of the first compiler (a program that translates source code into machine code). Notably, she is also credited with first using the term "computer bug" after a real bug (a moth) flew into a computer she was working on. Later in her career, she was one of the designers of COBOL, a programming language that is still in use today.

NASA and the Human Computers

The following women all worked on various NASA projects. Some even were hired as human computers. They were tasked with completing complex calculations by hand for all sorts of situations from wartime thrust-to-weight ratios to Apollo orbit trajectories. They all went on to have impressive careers in mathematics and computer science.

Annie Easley developed the energy analytics code used to analyze power technology including the technology that was used in battery technology for Centaur rockets and early hybrid vehicles

Katherine Johnson was a physicist, mathematician, and space scientist who provided the calculation for important missions like the first orbit of the Earth and the Apollo 11 moon landing.

Dorothy Vaughan was a mathematician who would eventually become the first African American supervisor of NACA (National Advisory Committee for Aeronautics which would later become NASA) and a FORTRAN expert programmer working on the Scout Launch Vehicle Program (a family of rockets that placed small satellites in orbit).

Mary Jackson was NASA's first Black female engineer. She worked on wind tunnel and flight experiments and would go on to earn NASA's most senior engineering title.

Melba Roy Mouton was a Head Mathematician at NASA working on Project Echo, the first experiment in passive satellite communication. At NASA, she wrote programs that calculated locations and trajectories of aircraft.

Evelyn Boyd Granville worked on multiple projects in the Apollo and Mercury programs for NASA. She worked on computer techniques related to concepts like celestial mechanics and trajectory computation.

Innovators in Modern Technology

Hedy Lamarr

Hedy Lamarr was born in 1914. A movie actress during the golden age of Hollywood, she was also a self-taught inventor. During the Second World War, she read about radio-controlled torpedoes which could potentially be jammed by enemy forces. She and a composer friend proposed and patented an idea for a frequency-hopping radio signal that used existing player piano technology. The principles of this work would eventually be used in familiar technologies like WiFI, Bluetooth, and GPS.

Guillermo Gonzalez Camarena

Guillermo Gonzalez Camarena was born in 1917. An electrical engineer, in 1940 he patented an adapter that let monochrome cameras use colors. This technology was one of the earliest forms of color television. Camarena's system would eventually be used by NASA for the Voyager mission and made color images of Jupiter possible.

Gerald (Jerry) Lawson

Jerry Lawson was born in 1940. Working as a semiconductor engineer for the Fairchild company, he worked on a team that developed the Fairchild Channel F, a color video game console that was designed to use interchangeable game cartridges. Previously, most game systems had built-in programming. He would later be dubbed the "father of the video game cartridge" for this work.

Mark E. Dean

Mark Dean was born in 1957. An inventor and computer scientist, he is the chief engineer of the IBM team that released the IBM personal computer. He holds three of the nine patents for the PC. He and his team also created the first gigahertz computer chip and he also helped develop the color PC monitor. Along with Dennis Moeller, he developed the Industry Standard Architecture (ISA) bus which was a precursor to modern bus structures like PCI and PCI express.

Clarence "Skip" Ellis

Clarence Ellis was born in 1943. He was a computer scientist and professor who pioneered in Computer Supported Cooperative Work and Groupware. In fact, while working at Xerox PARC, he and his team developed a groupware system called OfficeTalk. For the first time, this system allowed for collaboration from a distance using ethernet. He also focused on icon-based graphical user interfaces (GUIs) that have become prevalent in modern computing.

Gladys West

Gladys West was born in 1930. A mathematician, she was hired to work for the US Navy to more accurately model the shape of the Earth. She used algorithms to account for all sorts of variations in the shape of the Earth and her model would eventually be used as the basis for the Global Positioning System (GPS).

These individuals are a few notable examples, but this is by no means a complete list!

Supplemental Reading on Logic Gates

Logic Gates

Knowing how logic gates work is important to understanding how a computer works. Computers work by performing binary calculations. Logic gates are electrical components that tell a computer how to perform binary calculations. They specify rules for how to produce an electrical output based on one or more electrical inputs. Computers use these electrical signals to represent two binary states: either an "on" state or an "off" state. A logic gate takes in one or more of these binary states and determines whether to pass along an "on" or "off" signal.

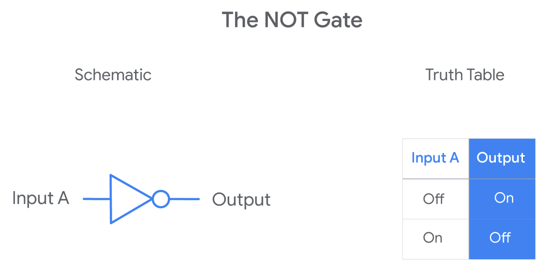
Several logic gates have been developed to represent different rules for producing a binary output. This reading covers six of the most common logic gates.

Six common logic gates

NOT gate

The NOT gate is the simplest because it has only one input signal. The NOT gate takes that input signal and outputs a signal with the opposite binary state. If the input signal is "on," a NOT gate outputs an "off" signal. If the input signal is "off," a NOT gate outputs an "on" signal. All the logic gates can be defined using a schematic diagram and truth table. Here's how this logic rule is often represented:

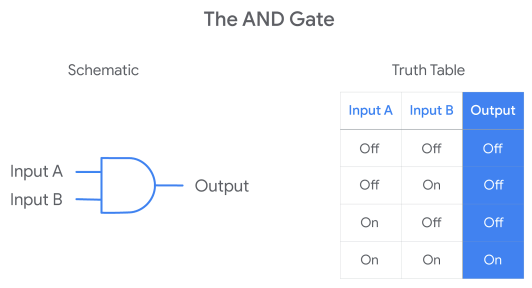
On the left, you have a schematic diagram of a NOT gate. Schematic drawings usually represent a physical NOT gate as a triangle with a small circle on the output side of the gate. To the right of the schematic diagram, you also have a "truth table" that tells you the output value for each of the two possible input values.



Input A	Output
Off	On
On	Off

AND gate

The AND gate involves two input signals rather than just one. Having two input signals means there will be four possible combinations of input values. The AND rule outputs an "on" signal only when both the inputs are "on." Otherwise, the output signal will be "off."



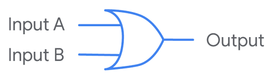
Input A	Input B	Output
Off	Off	Off
Off	On	Off
On	Off	Off
On	On	On

OR gate

The OR gate involves two input signals. The OR rule outputs an "off" signal only when both the inputs are "off." Otherwise, the output signal will be "on."

The OR Gate

Schematic



Truth Table

Input A	Input B	Output
Off	Off	Off
Off	On	On
On	Off	On
On	On	On

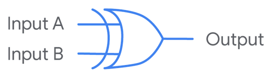
XOR Gate

The XOR gate also involves two input signals. The XOR rule outputs an "on" signal when *only one* (but *not both*) of the inputs are "on." Otherwise, the output signal will be "off."

The truth tables for XOR and OR gates are very similar. The only difference is that the XOR gate outputs an "off" when both inputs are "on" while the OR outputs an "on." Sometimes you may hear the XOR gate referred to as an "exclusive OR" gate.

The XOR Gate

Schematic



Truth Table

Input A	Input B	Output
Off	Off	Off
Off	On	On
On	Off	On
On	On	Off

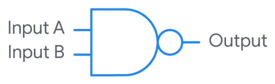
NAND gate

The NAND gate involves two input signals. The NAND rule outputs an "off" signal only when both the inputs are "on." Otherwise, the output signal will be "on."

If you compare the truth tables for the NAND and AND gates, you may notice that the NAND outputs are the opposite of the AND outputs. This is because the NAND rule is just a combination of the AND and NOT rules: it takes the AND output and runs it through the NOT rule! For this reason, you might hear the NAND referred to as a "not-AND" gate.

The NAND Gate

Schematic



Truth Table

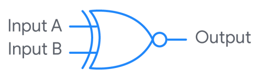
Input A	Input B	Output
Off	Off	On
Off	On	On
On	Off	On
On	On	Off

XNOR gate

Finally, consider the XNOR gate. It also involves two input signals. The XNOR rule outputs an "on" signal only when both the inputs are the same (both "On" or both "Off"). Otherwise, the output signal will be "off."

The XNOR Gate

Schematic



Truth Table

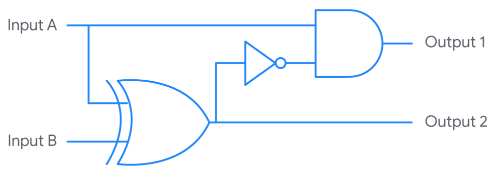
Input A	Input B	Output
Off	Off	On
Off	On	Off
On	Off	Off
On	On	On

The XNOR rule is another combination of two earlier rules: it takes the XOR output and runs it through the NOT rule. For this reason, you might hear the XNOR referred to as a "not-XOR" gate.

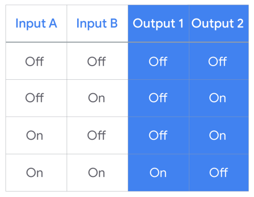
Combining gates (building circuits)

Logic gates are physical electronic components—a person can buy them and plug them into a circuit board. Logic gates can be linked together to create complex electrical systems (circuits) that perform complicated binary calculations. You link gates together by letting the output from one gate serve as an input for another gate or by using the same inputs for multiple gates. Computers are this kind of complex electrical system.

Here's a schematic drawing for a small circuit built with gates described above:



Here is the truth table for this circuit:



Input A	Input B	Output 1	Output 2
Off	Off	Off	Off
Off	On	Off	On
On	Off	Off	On
On	On	On	Off

This circuit uses three logic gates: an XOR gate, a NOT gate, and an AND gate. It takes two inputs (A and B) and produces two outputs (1 and 2). A and B are the inputs for the XOR gate. The output of that gate became the input of the NOT gate. Then, the output of the NOT gate became an input for the AND gate (with input A as the other). Output 1 is the output from the AND gate. Output 2 is the output from the XOR gate.

Key takeaways

Logic gates are the physical components that allow computers to make binary calculations.

- Logic gates represent different rules for taking one or more binary inputs and outputting a specific binary value ("on" or "off").
- Logic gates can be linked so that the output of one gate serves as the input for other gates.
- Circuits are complex electrical systems built by linking logic gates together. Computers are this kind of complex electrical system.

Binary Conversion

Decimal values, binary values, and characters are all used to communicate information. Computers receive and communicate information with binary values, so the binary system shapes the rules and conventions of how computers interact with one another. Being able to convert binary values to decimal values or characters will help you better understand IT infrastructure and computer networking. In this reading, you'll learn more about converting between decimal values, binary values, and characters. You'll also practice using a binary conversion table.

Use a table to convert between decimal and binary

By convention, decimal numbers are represented with 8 bits (1 byte) in binary. Each bit is either a 0 or a 1, so 28 = 256 decimal numbers can be represented with 1 byte. Additionally, each bit represents a specific decimal value based on its order in the byte. The 1st (leftmost) bit is 128, and each bit after that is half the value of the previous one.

You can use a conversion table, like the one that follows, to convert from decimal to binary, and vice versa:

	1st bit	2nd bit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit
Decimal value	128	64	32	16	8	4	2	1
Off or on	<fill in>	<fill in>	<fill in>	<fill in>	<fill in>	<fill in>	<fill in>	<fill in>

In this table, the "Decimal value" row displays the decimal value of each bit in the byte. For example, the 3rd bit has a value of 32 and the 8th bit has a value of 1. To use this table for a conversion, the "Off or on" row is filled to indicate if a bit is off (0) or on (1).

Then, the numbers in the "Decimal value" rows that have a 1 in the "Off or on" row of the same column are added together to get the decimal value of the byte.

This might seem complicated, but it just takes some practice! In the next sections, you'll use this table to convert from binary to decimal values and from decimal to binary values.

Convert from binary to decimal values

To use the table to convert from binary to decimal values, enter the byte you want to convert into the "Off or on" row. For example, to convert the byte 10011101 to a decimal value, fill the "Off or on" row with the values of each bit in the byte, like this:

	1st bit	2nd bit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit
Decimal value	128	64	32	16	8	4	2	1
Off or on	1	0	0	1	1	1	0	1

In this example, 128 + 16 + 8 + 4 + 1 = 157. Therefore, the decimal value represented by the binary number 10011101 is 157.

Convert from decimal to binary values

To use this table to convert from a decimal value to a binary value, put 0s and 1s in the “Off or on” row of the table so that the sum of the decimal values of any columns that contain a 1 in the “Off or on” row add up to the decimal value.

For example, to convert the decimal value 87 to binary, you’d fill out the table like this:

	1st bit	2nd bit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit
Decimal value	128	64	32	16	8	4	2	1
Off or on	0	1	0	1	0	1	1	1

The sum of 64 + 16 + 4 + 2 + 1 is 87, so the binary value that represents 87 is 01010111.

Try it yourself!

Now, practice on your own by completing the following practice problems:

1. Use the binary conversion table to convert the binary value 00010011 into a decimal value.
2. Use the binary conversion table to convert the decimal value 179 into a binary value.

Navigate to the end of this reading to check your answers.

Character encoding: From binary values to characters

As you learned earlier in this lesson, character encoding assigns binary values to characters so that humans can read them. The American Standard Code for Information Interchange (ASCII) was the first character encoding standard used. It uses one byte to represent each character in the English alphabet, digits, and punctuation. Each byte maps to a specific character, so ASCII can only represent 256 characters.

The following table displays the ASCII table for the first 5 lowercase letters in the English alphabet:

Binary value	Decimal value	Character
01100001	97	a
01100010	98	b
01100011	99	c
01100100	100	d
01100101	101	e

UTF-8 is a newer standard that uses the same ASCII character encodings but allows characters to be represented with more than one byte. This allows many more characters–and even emojis–to be represented with binary.

Key takeaways

Computers communicate using binary, so it is important for IT support specialists to understand how binary works and to be able to convert binary values into both decimal values and characters. You can use a binary conversion table to help you convert between decimal and binary values. You can use an ASCII or UTF-8 table to convert from binary or decimal values into characters.

Practice exercise answers

1. Use the binary conversion table to convert the binary value 00010011 into a decimal value. The decimal value of 00010011 is 19.
2. Use the binary conversion table to convert the decimal value 179 into a binary value. The binary value of 179 is 10110011.

Supplemental Reading for CPUs

CPU cache and overclocking

In this reading, you will learn about the various levels of cache for central processing units (CPUs) and how a CPU processes and executes instructions. Additionally, you will learn about overclocking CPUs to maximize processing speeds. IT Support professionals may use this information when purchasing, allocating, and/or configuring high-performance servers.

Cache

You may already be familiar with the term “cache”. In computer jargon, cache (pronounced “cash”) refers to a small amount of recently used data that is stored either on hardware or in software. The first time data is accessed, both the initial request for the data and the reply containing the data pass through multiple points on their journey. Depending on several variables, these points might include I/O devices, motherboard busses, RAM, cables, hard drives, applications, networks, the internet, cloud platforms, and more. If a computer needed to use these full paths everytime it tried to access data, the entire transaction could take a relatively long time. Cache speeds up this process by holding a local copy of the most recently accessed data in temporary storage.

CPU cache

CPUs use a system of cache storage to help them quickly access data. A CPU cache is normally stored inside each core of the CPU. Older computers might store CPU cache in a transistor chip that is attached to the motherboard, along with a high-speed bus connecting the chip to the CPU.

CPU levels of cache

There are three levels of CPU cache memory:

- Level 3 cache: L3 cache is the largest and slowest of CPU cache. However, it is often twice as fast as RAM. L3 is the first CPU cache location to store data after it is transferred from RAM. L3 cache is often shared by all of the cores in a single CPU.
- Level 2 cache: L2 cache holds less data than L3 cache, but it has faster access speeds. L2 holds a copy of the most recently accessed data that is not currently in use by the CPU. Each CPU core normally has its own L2 cache.
- Level 1 cache: L1 cache is the fastest and smallest of the three CPU cache levels. L1 holds the data currently in use by the CPU. Each CPU core usually has its own L1 cache.

Overclocking a cpu

Overclocking a CPU sets it to run at a higher CPU clock frequency rate than the manufacturer’s original specifications. For example, if a processor is labeled as having a 3.2 GHz base frequency rate, it may be possible to overclock the CPU to run at 3.5 GHz. Achieving a higher CPU clock frequency rate means the CPU can process a higher volume of instructions per nanosecond, resulting in faster performance. A computer user might want to overclock their CPU to improve sluggish speeds when performing processor-intensive tasks, like video editing or gaming.

Overclocking a CPU’s frequency involves three variables:

- The base CPU clock frequency, often measured in GHz.
- The core frequency, which is calculated by multiplying the base frequency by the CPU core multipliers.
- The core voltage, which needs to be increased in small increments to meet the increasing power demand of the CPU during the overclocking process.

Warnings on overclocking

Overclocking the CPU can damage the computer if not configured properly. Operating a CPU at a higher speed can overheat the
CPU and surrounding hardware, which can cause the computer system to fail. Additionally, overclocking the CPU can shorten the overall lifespan of the computer and void the computer's warranty. It is better to avoid overclocking the CPU and instead purchase the appropriate CPU speed necessary to meet computing demands.

How to overclock a CPU safely

As an IT Support professional, you may be asked to overclock a CPU. There are steps you should follow to do this as safely as possible. Always make sure that the requestor understands the risks of overclocking before agreeing to perform this procedure.

1. Check if overclocking is supported: First, make sure the CPU is a model that is unlocked for overclocking. Not all CPUs can support overclocking, including most laptop CPUs. Check the CPU manufacturer's documentation to determine if overclocking is possible for the CPU model. Both Intel and AMD provide overclocking guides and tools for supported CPU models (see below for links to these guides). Additionally, check the documentation for the computer's motherboard model to ensure that it can support an overclocked CPU.
2. Clean the inside of the computer: Turn off and unplug the computer. While wearing an anti-static wristband, open the computer and use compressed air to remove any dust build-up that has accumulated. It is especially important to remove any dust from around the CPU, fans, and intake vents.
3. Ensure an appropriate CPU cooler is installed (critical): If the computer has a stock CPU cooler, it is most likely insufficient for cooling an overclocked CPU. Replace the stock CPU cooler with an advanced cooling system, like a liquid cooling system.
4. Follow the manufacturer's instructions for overclocking the CPU: Using the detailed instructions from the manufacturer (see below for links to Intel and AMD's guides):
    a. Use benchmarking software to establish a baseline for the normal performance of the computer.
    b. Set each CPU core multiplier to the value of the lowest multiplier using either the manufacturer's overclocking software (recommended) or the BIOS. Then reboot the computer.
    c. Increase each CPU core multiplier by 1 to increase the CPU frequency.
    d. Test each increase for stability using the testing utility provided by the manufacturer.
        - Fix any problems flagged by the testing tools, especially temperature alerts. If the system becomes too unstable, roll back to the last frequency that produced a stable performance and stop overclocking the CPU.
        - If the voltage appears to become insufficient to support the new frequency, increase the voltage by 0.05V. Do not increase the voltage above 1.4V without specialized cooling hardware.
        - If the computer freezes or crashes, it has either become completely unstable or the CPU is not getting enough voltage to support the overclocked frequency. Use the BIOS to return to the last stable frequency or increase the voltage in 0.01V increments until stable.
    e. If stable, reboot the computer before attempting the next increase.

Resources for more information

- Intel: Overclocking: Maximize Your Performance - Intel's all-inclusive guide to overclocking CPU, RAM, and motherboard. The site also provides utility tools for fine-tuning overclock performance and lists Intel CPU models that support overclocking.
- AMD: AMD Ryzen™ Master Utility for Overclocking Control - AMD's toolkit for overclocking Ryzen processors. Note that overclocking support for non-Ryzen models is no longer recommended by AMD.
- AMD: Ryzen™ Processor Overclocked Memory Compatibility List - List of AMD Ryzen CPU models that support overclocking.
- AMD: How to Overclock Your AMD Ryzen CPU - Instructions for overclocking AMD Ryzen CPUs from PC Magazine.

Supplemental Reading for Data Storage

Data Storage Measurements
In this reading, you will learn about the different names for measurements of data storage capacities and file sizes. Data storage capacity increases in step with the evolution of computer hardware technology. Larger storage capacities allow for dynamic growth in file sizes. These advances make it possible for companies like Netflix and Hulu to store thousands of feature-length films in high video quality formats.

There are standardized sets of terms used to name the ever expanding sizes of data storage and files. For example, the common terms used to describe file sizes and hard drive storage capacity include: bytes, kilobytes, megabytes, gigabytes, and terabytes. However, if you are a computer engineer, you might use a different set of terms.

Data storage measurement nomenclature

- Decimal nomenclature: kilobyte, megabyte, gigabyte, terabyte, petabyte, exabyte, zettabyte, yottabyte

The decimal naming system for computer storage uses the metric system of prefixes from the International System of Units: kilo, mega, giga, tera, peta, exa, zetta, and yotta. These prefixes may also be referred to as the decimal system of prefixes. The metric/decimal nomenclature represent a base-10 approximation of the actual amount of data storage bytes. The metric system prefixes were selected to simplify the marketing of computer products.

Decimal			
Name	Abbrv.	Value in base 10	Full value in bytes
kilobyte	KB	10^3	1,000
megabyte	MB	10^6	1,000,000
gigabyte	GB	10^9	1,000,000,000
terabyte	TB	10^{12}	1,000,000,000,000
petabyte	PB	10^{15}	1,000,000,000,000,000
exabyte	EB	10^{18}	1,000,000,000,000,000,000
zettabyte	ZB	10^{21}	1,000,000,000,000,000,000,000
yottabyte	YB	10^{24}	1,000,000,000,000,000,000,000,000

- Binary nomenclature: kibibyte, mebibyte, gibibyte, tebibyte, pebibyte, exbibyte, zebibyte, yobibyte

The binary naming system is a standard set by the International Organization for Standardization (ISO) in partnership with the International Electrotechnical Commission (IEC). The ISO 80000 and IEC 80000 guides to units of measurement define the International System of Quantities (ISQ). The prefixes kibi-, mebi-, gibi, -tebi-. pebi-, exbi-, zebi-, and yobi- were created by the IEC organization. They are a blend of the first two letters of the metric prefix fused with the first two letters of the word "binary" (example: megabyte + binary + byte= mebibyte).

Binary measurements of computer data are more accurate than decimal system measurements.

Binary			
Name	Abbrv.	Value in base 2	Full value in bytes
kibibyte	KiB	2^{10}	1,024
mebibyte	MiB	2^{20}	1,048,576
gibibyte	GiB	2^{30}	1,073,741,824
tebibyte	TiB	2^{40}	1,099,511,627,776
pebibyte	PiB	2^{50}	1,125,899,906,842,620
exbibyte	EiB	2^{60}	1,152,921,504,606,850,000
zebibyte	ZiB	2^{70}	1,180,591,620,717,410,000,000
yobibyte	YiB	2^{80}	1,208,925,819,614,630,000,000,000

While decimal nomenclature is commonly used to market computers and computer parts to the general public, binary nomenclature is often used in computer engineering for numerical accuracy.

– – Continued – –

Quantities of storage measurements

As data storage grows, the need for new terminology to describe the exponentially larger byte quantities grows too. The current byte nomenclature, mathematical representations, and storage capacities are as follows:

- **One bit**: Also called a binary digit, bits store an electric signal as 1. The absence of an electric signal is stored as 0, which is also the default value of a bit. One bit can store only one value, either 1 or 0. These two possible values are the basis of the binary number system (base-2) that computers use. All numbers in a base-2 system increase exponentially as powers of 2.

- **One byte**: One byte stores eight bits of ones and zeros that translate to a symbol or basic computer instruction. Examples: 01101101 is the byte that translates to the letter "m." The byte 01111111 tells the computer to delete the character to the right of the cursor.

- **One kilobyte (1 KB)**:
  - Kilobyte (KB) decimal format: 103 = 1,000 bytes
  - Kibibyte (KiB) binary format: 210 = 1,024 bytes
  - Decimal inaccuracy: Off by -2.4% or -24 bytes
  - Name origin: "Kilo-" is a French derivation from the Ancient Greek word for "thousand" A kilobyte is one thousand bytes.
  - 1 KB can hold: A short text file or a small icon as a 16x16 pixel .gif file.

- **One megabyte (1 MB)**:
  - Megabyte (MB) decimal format: 106 = 1,000,000 bytes
  - Mebibyte (MiB) binary format: 220= 1,048,576 bytes
  - Decimal inaccuracy: Off by -4.9% or -48,576 bytes
  - Name origin: "Mega-" is derived from the Ancient Greek word for "large." A megabyte is a large number of bytes.
  - 1 MB can hold: Approximately one minute of music in a lossless .mp3 format or a short novel.

- **One gigabyte (1 GB)**:
  - Gigabyte (GB) decimal format: 109 = 1,000,000,000 bytes
  - Gibibyte (GiB) binary format: 230 = 1,073,741,824 bytes
  - Decimal inaccuracy: Off by -7.4% or -73,741,824 bytes
  - Name origin: "Giga-" is derived from the Ancient Greek word for "giant." A gigabyte is a giant number of bytes.
  - 1 GB can hold: Between 2.5-3 hours of music in .mp3 format or 300 high-resolution images.

- **One terabyte (1 TB)**:
  - Terabyte (TB) decimal format: 1012 = 1,000,000,000,000 bytes
  - Tebibyte (TiB) binary format: 240 = 1,099,511,627,776 bytes
  - Decimal inaccuracy: Off by -10.0%
  - Name origin: "Tera-" is a shortened form of "tetra-", which was derived from the Ancient Greek word for the number four. The 1012 decimal format can also be written as 10004 (one-thousand to the 4th power). "Tera-" in Ancient Greek means "monster." You might think of the word "terabyte" as a monstrously large number of bytes.
  - 1 TB can hold: Approximately 200,000 songs in .mp3 format or 300 hours of video.

- **One petabyte (PB)**:
  - Petabyte (PB) decimal format: 1015 = 1,000,000,000,000,000 bytes
  - Pebibyte (PiB) binary format: 250 = 1,125,899,906,842,624 bytes
  - Decimal inaccuracy: Off by -12.6%
  - Name origin: "Peta-" is derived from the Ancient Greek word "penta" meaning five. The 1015 decimal format can also be written as 10005 (one-thousand to the 5th power).
  - 1 PB can hold: The content from 1.5 million CD-ROM discs or 500 billion pages of text.

- **One exabyte (EB)**:
  - Exabyte (EB) decimal format: 1018 = 1,000,000,000,000,000,000 bytes
  - Exbibyte (EiB) binary format: 260 = 1,152,921,504,606,846,976 bytes
  - Decimal inaccuracy: Off by -15.3%
  - Name origin: "Exa-" was derived from the Ancient Greek word for six. The 1018 decimal format can also be written as 10006 (one-thousand to the 6th power).
  - 1 EB can hold: Approximately 11 million movies in 4k video resolution or 3,000 copies of the entire United States Library of Congress.

- **One zettabyte (ZB)**:
  - Zettabyte (ZB) decimal format: 1021 = 1,000,000,000,000,000,000,000 bytes
  - Zebibyte (ZiB) binary format: 270 = 1,180,591,620,717,411,303,424 bytes

    - Decimal inaccuracy: Off by -18.1%
    - Name origin: “Zetta” was derived from the Latin word “septem” which means seven. The 1021 decimal format can also be written as 10007 (one-thousand to the 7th power).
    - 1 ZB can hold: Seagate reports one zettabyte can hold 30 billion movies in 4k video resolution.

- **One yottabyte (YB)**:
    - Yottabyte (YB) decimal format: 1024 = 1,000,000,000,000,000,000,000,000 bytes
    - Yobibyte (YiB) binary format: 280 = 1,208,925,819,614,629,174,706,176 bytes
    - Decimal inaccuracy: Off by -20.9%
    - Name origin: “Yotta” is Ancient Greek for eight. The 1024 decimal format can also be written as 10008.
    - 1 YB can hold: In 2011, a cloud storage company estimated that one yottabyte could hold the data of one million data centers.

Common Scripting Solutions

In this reading, you will learn about a variety of scripting languages, their uses, and their risks. As an IT Support professional, you may need to automate routine tasks. For example, you might want to automate a backup of company data that runs every night. You might also need to automate high volume tasks, like changing security access settings on thousands of files. Scripting is a common tool used for automation. This tool can help IT Support staff save time and resources in a busy enterprise work environment.

Scripting languages

There are many scripting languages available to use for a variety of tasks in different operating system environments. Most scripts are written in command line environments.

Scripting languages for **Windows** environments:

- **PowerShell (.ps1)** - Windows PowerShell is among the most common command line scripting tools used in Windows environments. PowerShell is built on the .NET platform and employs many of the same elements that programming languages do. PowerShell scripts are used for building, testing, and deploying solutions, in addition to automating system management.
- **Batch scripts (.bat)** - Batch scripts, also called batch files, have been around since the early days of MS DOS and OS/2. Batch files can execute simple tasks, like calling a set of programs to run when a computer boots up. This type of script could be useful in setting up employees' workspaces when they power on their computers.
- **Visual Basic Script (.vbs)** - Visual Basic Script is an older scripting language. It has reached its end of life for Microsoft support and has been replaced by PowerShell scripts. However, as an IT professional, you may encounter .vbs scripts on some legacy systems.

Scripting languages for **Linux** and **Unix** environments:

- **Shell script (.sh)** - Shell scripting languages, like Bash, are used in Unix or Linux environments. The scripts are often used to manipulate files, including changing file security settings, creating, copying, editing, renaming and deleting files. They can also be used to execute programs, print, navigate the operating system, and much more. The scripts run in command-line interpreter (CLI) shells, such as the Bourne shell, Bourne Again SHell (Bash), C shell, and Korn (KSH) shell.

Programming languages that can be used for **scripting**:

- **JavaScript (.js)** - JavaScript the most used programming language in the world. It is a lightweight language that is used for scripting in web development, mobile and web apps, games, and more. It can also be used to develop software and automate web server functions.
- **Python (.py)** - Python is a user-friendly programming language that can perform advanced tasks and import modules from libraries specially designed for automation scripts.

Scripting uses - finding the right tool for the job

- **Basic automation**: Python is an excellent script for automation. It's one of the most commonly used, with many available automation libraries.
- **Restarting machines**: Many power users use PowerShell (.ps1) scripts to restart machines (Windows). For Linux machines, they can use .sh (shell) scripts.
- **Mapping network drives**: In the past, mapping network drives was accomplished with .bat or .vbs scripts. However, PowerShell scripts are most commonly used to map drives in Windows environments today. For Linux users, shell scripts can be used for this purpose.
- **Installing applications**: Batch files and shell scripts are often used for automated software installation.
- **Automated Backups**: Windows PowerShell and Linux/Unix shell scripts can automate backups.
- **Gathering of information and data**: Python is a popular choice for gathering data. Python has many available libraries to help with this task.
- **Initiating Updates**: Powershell and shell scripts can be used for initiating updates in Windows and Linux, respectively.

– – Continued – –

Security risks of using scripts

IT Support professionals need to be very careful when using scripts, especially with prewritten scripts copied or downloaded from the internet. Some of the security risks of using scripts could include:

- Unintentionally introducing malware: As an IT Support professional that is new to scripting, you may try to search the internet for assistance in writing scripts. In your search, you might find a script online for a task that you want to automate. It's tempting to save time and effort by downloading the script and deploying it in your network environment. However, this is dangerous because scripts authored by an unverified source could potentially contain malware. Malicious scripts could have the power to delete files, corrupt data and software, steal confidential information, disable systems, and even bring down an entire network. Malicious scripts can create security weaknesses for the purpose of creating entry points for cybercriminals to penetrate networks. Scripts could also introduce ransomware attacks, which often works by encrypting file systems and then selling the decryption keys for ransom.
- Inadvertently changing system settings: Scripts are powerful tools for changing system settings. Using the wrong script can cause the user to inadvertently configure harmful settings. For example, one minor typo in a shell script that sets file permission security in Linux could make confidential files accessible to the world.
- Browser or system crashes due to mishandling of resources: Mishandling resources can lead to program crashes in the browser or cause the entire computer to crash. For example, directing too much memory to the browser can overload the computer system.

Key takeaways

A basic knowledge of scripting is an important tool for IT professionals. You may need to improve workflow efficiency by automating basic functions with a scripting language. Some common scripting languages include:

- Windows environments: batch scripts (.bat), Powershell (.ps1), Visual Basic Script (.vbs)
- Linux/Unix environments: shell scripts (.sh)
- Most OS environments: javascript (.js), Python (.py)

Scripts have multiple helpful uses, such as:

- Basic Automation
- Restarting Machines
- Remapping Network Drives
- Installing Applications
- Automating Backups
- Gathering of information/ data
- Initiating Updates

There are risks in using scripts, including:

- Unintentionally introducing malware
- Inadvertently changing system settings
- Browser or system crashes due to mishandling of resources

Resources for more information
For more information about scripting languages, please visit:

- 14 Top Scripting Languages You Can Learn