A terminal is at the end of an electric wire, a shell is the home of a turtle, tty is a strange abbreviation and a console is a kind of cabinet.

Well, etymologically speaking, anyway.

In Unix terminology, the short answer is that

* terminal = tty = text input/output environment
* console = physical terminal
* shell = command line interpreter

Console, terminal and tty are closely related. Originally, they meant a piece of equipment through which you could interact with a computer: in the early days of Unix, that meant a [teleprinter](http://en.wikipedia.org/wiki/Teleprinter)-style device resembling a typewriter, sometimes called a teletypewriter, or “tty” in shorthand. The name “terminal” came from the electronic point of view, and the name “console” from the furniture point of view. Very early in Unix history, electronic keyboards and displays became the norm for terminals.

In Unix terminology, a **tty** is a particular kind of [device file](http://en.wikipedia.org/wiki/Device_file) which implements a number of additional commands ([ioctls](http://en.wikipedia.org/wiki/Ioctl" \l "Terminals)) beyond read and write. In its most common meaning, **terminal** is synonymous with tty. Some ttys are provided by the kernel on behalf of a hardware device, for example with the input coming from the keyboard and the output going to a text mode screen, or with the input and output transmitted over a serial line. Other ttys, sometimes called **pseudo-ttys**, are provided (through a thin kernel layer) by programs called [**terminal emulators**](http://en.wikipedia.org/wiki/Terminal_emulator), such as [Xterm](http://en.wikipedia.org/wiki/Xterm) (running in the [X Window System](http://en.wikipedia.org/wiki/X_Window_System)), [Screen](http://en.wikipedia.org/wiki/Gnu_screen) (which provides a layer of isolation between a program and another terminal), [Ssh](http://en.wikipedia.org/wiki/Secure_shell) (which connects a terminal on one machine with programs on another machine), [Expect](http://en.wikipedia.org/wiki/Expect) (for scripting terminal interactions), etc.

The word terminal can also have a more traditional meaning of a device through which one interacts with a computer, typically with a keyboard and display. For example an X terminal is a kind of [thin client](http://en.wikipedia.org/wiki/Thin_client), a special-purpose computer whose only purpose is to drive a keyboard, display, mouse and occasionally other human interaction peripherals, with the actual applications running on another, more powerful computer.

A **console** is generally a terminal in the physical sense that is by some definition the primary terminal directly connected to a machine. The console appears to the operating system as a (kernel-implemented) tty. On some systems, such as Linux and FreeBSD, the console appears as several ttys (special key combinations switch between these ttys); just to confuse matters, the name given to each particular tty can be “console”, ”virtual console”, ”virtual terminal”, and other variations.

See also [Why is a Virtual Terminal “virtual”, and what/why/where is the “real” Terminal?](https://askubuntu.com/q/14284/1059).

A [**shell**](http://en.wikipedia.org/wiki/Shell_%28computing%29) is the primary interface that users see when they log in, whose primary purpose is to start other programs. (I don't know whether the original metaphor is that the shell is the home environment for the user, or that the shell is what other programs are running in.)

In unix circles, **shell** has specialized to mean a [command-line shell](http://en.wikipedia.org/wiki/Shell_%28computing%29#Text_.28CLI.29_shells), centered around entering the name of the application one wants to start, followed by the names of files or other objects that the application should act on, and pressing the Enter key. Other types of environments don't use the word “shell”; for example, window systems involve “[window managers](http://en.wikipedia.org/wiki/Window_manager)” and “[desktop environments](http://en.wikipedia.org/wiki/Desktop_environment)”, not a “shell”.

There are many different Unix shells. Popular shells for interactive use include [Bash](http://en.wikipedia.org/wiki/Bash_(Unix_shell)) (the default on most Linux installations), [zsh](http://en.wikipedia.org/wiki/Zsh) (which emphasizes power and customizability) and [fish](http://en.wikipedia.org/wiki/Friendly_interactive_shell) (which emphasizes simplicity).

Command-line shells include flow control constructs to combine commands. In addition to typing commands at an interactive prompt, users can write scripts. The most common shells have a common syntax based on the [Bourne\_shell](http://en.wikipedia.org/wiki/Bourne_shell). When discussing “**shell programming**”, the shell is almost always implied to be a Bourne-style shell. Some shells that are often used for scripting but lack advanced interactive features include [the Korn shell (ksh)](http://en.wikipedia.org/wiki/Korn_shell) and many [ash](http://en.wikipedia.org/wiki/Almquist_shell) variants. Pretty much any Unix-like system has a Bourne-style shell installed as /bin/sh, usually ash, ksh or bash.

In unix system administration, a user's **shell** is the program that is invoked when they log in. Normal user accounts have a command-line shell, but users with restricted access may have a [restricted shell](http://en.wikipedia.org/wiki/Restricted_shell) or some other specific command (e.g. for file-transfer-only accounts).

The division of labor between the terminal and the shell is not completely obvious. Here are their main tasks.

* Input: the terminal converts keys into control sequences (e.g. Left → \e[D). The shell converts control sequences into commands (e.g. \e[D → backward-char).
* Line editing, input history and completion are provided by the shell.
  + The terminal may provide its own line editing, history and completion instead, and only send a line to the shell when it's ready to be executed. The only common terminal that operates in this way is M-x shell in Emacs.
* Output: the shell emits instructions such as “display foo”, “switch the foreground color to green”, “move the cursor to the next line”, etc. The terminal acts on these instructions.
* The prompt is purely a shell concept.
* The shell never sees the output of the commands it runs (unless redirected). Output history (scrollback) is purely a terminal concept.
* Inter-application copy-paste is provided by the terminal (usually with the mouse or key sequences such as Ctrl+Shift+V or Shift+Insert). The shell may have its own internal copy-paste mechanism as well (e.g. Meta+W and Ctrl+Y).
* [Job control](http://en.wikipedia.org/wiki/Job_control) (launching programs in the background and managing them) is mostly performed by the shell. However, it's the terminal that handles key combinations like Ctrl+C to kill the foreground job and Ctrl+Z to suspend it.

++++++++++++++++++++++++++++++++++++

A **terminal** or a **console** is a piece of hardware, using which a user can interact with a host. Basically a keyboard coupled with a text screen.  
Nowadays nearly all terminals and consoles represent "virtual" ones.

The file that represents a terminal is, traditionally, called a **tty** file. If you look under the "/dev" directory of a UNIX system, you'll find a lot of **tty** files connected to virtual consoles (e.g. tty1 on linux), virtual terminals (e.g. pts/0) or physically connected hardware (e.g. ttyS0 is the physical serial terminal, if any, attached on first serial port of the host).

A **console** must be a piece of hardware physically connected to (or part of) the host. It has a special role in the system: it is the main point to access a system for maintenance and some special operation can be done only from a console (e.g. see single user mode). A **terminal** can be, and usually is, a remote piece of hardware.

Last, but not the least, a **shell** is a special program that interacts with a user through a **controlling tty** and offers, to the user, the way of launching other programs (e.g. bash, csh, tcsh).

A **terminal emulator** is a program that emulates a physical terminal (e.g. xterm, gnome-terminal, minicom).

So when you look to a "text window" on your linux system (under X11) you are looking to: a *terminal emulator*, connected to a *virtual terminal*, identified by a *tty* file, inside which runs a *shell*.