ADMINISTRIVIA: NO CLASS NEXT WEEK (May 2nd, 2018)

Debugging (moreso troubleshooting) live software is inevitable. Just make sure you set yourself (or the operations personnel) up for success! Even though you've accounted for edge cases, race conditions, efficiency and scale, it's still likely there will be something unexpected that will cause a need to troubleshoot. One of the best tools for this is logging.

Logging levels: Logging is generally divided into level, that represent how "detailed" they are. Commonly you'll have at least 3-4 levels, perhaps:

"Debug":	Lots of logging, very detailed.
"Info":	Moderate amount of logging, normal operation is logged as sort of a "progress" output
"Warning":	("Yellow") Possibly a problem, at least something someone should look at. (non-fatal)
"Fatal" / "Emergency":	("Red") Really bad issue. Program may terminate after a fatal error.

Logging is NOT "free", it consumes resources: CPU, Network, I/O. Therefore, in general, you won't be logging at Debug (or possibly even Info) level on production systems. (See pitfalls, below ;))

Some companies will add instrumentation to certain function calls that will help gather certain data, usable later. This generally requires a bit of an investment on the infrastructure side, and also an awareness that you are probably sucking an extra 1-5% of performance off the top. Examples:

- eBay's "CAL" system (Centralized Application/Activity Logging). Back around 2006-2007 it was generating about 2TB of logs per day (north of 10TB a day as of 2014!)
- Blizzard's (battle.net's) database call timing (stored procedures)
- Blizzard's (battle.net's) centralized logging (custom log servers that buffer data and guarantee delivery to the central server)

Dangerous pitfalls: (1) Over-logging, which will waste (or possibly completely consume) one of the mentioned resources (CPU, Network, Disk I/O). (Happened at Blizzard... also happened w/ monitoring systems (SNMP) way back in the 90's at Globalcenter). (2) Jumbled or interleaved logging, which will make trying to track a problem nearly impossible. If you have multiple log lines for a given "event", try to ensure that there is some unique identifier that correlates them.

Quitting and restarting a process that is being troubleshot is often not an option, so it's useful to have other ways to increase logging (sending the process a signal to either directly adjust logging, or to force it to re-read a configuration file that sets logging levels).

Sometimes programs will actually behave differently when in "debug mode" (additional logging) due to how and when (and which) functions are being called. This is beyond aggravating when trying to track down certain issues/bugs, since they often won't occur once you enable logging!

"Passive" logging (more like just exposing the data in some way, not necessarily directly logging it) can be useful for things like web pages that are generated on-the-fly (e.g. hover/alt tags like FB used to do, or using HTML comments)

Logging for tracking purposes (or things like counting usage, etc) can be an enormous waste of resources, so often you'll want to make this a *sampling* of the total traffic... only log 1 out of every 100 or 10,000 operations. (But avoid "magic numbers" in your code, if you can). There can sometimes be some pitfalls here, unless you can make the random sampling based on something static (e.g. a person's user ID), because of possible related transactions that are logged in one place, but not logged in another place.