## <https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html>

<https://betterstack.com/community/guides/logging/python/python-logging-best-practices/>

## Logging Notes:

* Logging should always be included for any security events
  + UI login
  + Anytime SSH is used in scripts
  + When API calls are made
  + When API responses are received
  + When configurations change from the ATS UI
* Logging may also be useful for performance monitoring
* Avoid the root logger in python. It is instead recommended to create a logger for each module or component in the application.
  + Allows for individual log control
  + Easy to separate log data for analysis
* To create logger for each module in python:
  + Logger = logging.getLogger(“my\_module”)
  + Once this is done, can use standard logging methods
    - debug()
    - info()
    - warning()
    - error()
    - critical()
  + Make sure to set logger.propagate = False
    - Prevents duplication
* Create a separate module for logging config
* Define log format, and output destination
  + {“asctime”: “2024-01-01 00:00:00,000”, “levelname”: “INFO”, “message”: “An info”}
* Import module into the application code - allows to use the same log settings across all modules
* LOG LEVELS are used to indicate severity
  + **CRITICAL:** Shows that the error is very serious and requires urgent attention. Application may be unable to continue running.
  + **ERROR:** Shows an error or failure to perform some task or function. Ex: database errors or http request failures
  + **WARNING:** Shows information that indicates that something unexpected happened, or there is a possibility of a future problem. Ex: database storage low
  + **INFO:** General information about the app to ensure it runs as expected
  + **DEBUG:** Detailed information, relevant for diagnosing problems.
* Setting log levels allows control over which messages display in output. Useful for different environments

**Sensitive data censor example:**



* Rotate (AKA, delete old and create new) log files periodically
  + Can do either: Time-Based rotation, size-based rotation, hybrid rotation
* Centralise logs onto a single server