codecademy

Logging

Setting the Logging Level

Setting the logging level for the logger object will allow all messages at and above that level to be produced in the output stream. The logging level can be set by using `setLevel()`. If no level is manually set, the default level `logging.WARNING` is used.

```
import logging

logger.setLevel(logging.INFO)
```

Formatting Options for Logging

Formatting of logged messages can be changed by using the `Formatter` class. Information like timestamps, function names, and line numbers can be included in the logged message.

```
import logging

formatter = logging.Formatter('[%
(asctime)s] %(message)s')
```

Logging to File and Console

Log messages can be directed to both files and the console by adding `FileHandler` and `StreamHandler` handler objects to the logger.
While logging to console is helpful if needing to review debugging log messages in real-time, logging to a file allows for the logged messages to exist well after the program execution has occurred.

```
import logging
import sys

file_handler
= logging.FileHandler("program.log")
stream_handler
= logging.StreamHandler(sys.stdout)
```

Configuring the Logger with basicConfig

For a simple, one-liner configuration of the logger, there is a `basicConfig()` method that will allow for adding handlers, setting formatting options for the logged messages, and setting the log level for the logger.

```
import logging

logging.basicConfig(filename='program.log'
, level=logging.DEBUG, format= '[%
(asctime)s] %(levelname)s - %(message)s')
```

Using the Logging Module

Logging messages and exceptions is important for debugging applications and for logging important information that generates during execution. The proper method for logging these messages is the `logging` module.

```
import logging
```

Logging Levels

There are six logging levels associated with the logging module:

- NOTSET, which has a numeric value of 0
- DEBUG, which has a numeric value of 10
- INFO, which has a numeric value of 20
- WARNING, which has a numeric value of 30
- ERROR, which has a numeric value of 40
- CRITICAL, which has a numeric value of 50

```
import logging

logging.DEBUG
logging.INFO
logging.WARNING
logging.ERROR
logging.CRITICAL
logging.NOTSET
```