Logging

Logging is useful to track the error or exception or information. It also helps in debugging.

We use Logging Module to log the error.

Syntax:-

```
import logging
from logging import *
```

basicConfig (**kwargs) Method

This method is used to config the logging System.

Syntax:-

basicConfig(**kwargs)

- filename – It specifies that a FileHandler be created, using the specified filename, rather than a StreamHandler.
- filemode - If filename is specified, open the file in this mode. Defaults to 'a’. We can write ‘w’
- level - Set the root logger level to the specified level.
- format - Use the specified format string for the handler.
- datefmt - Use the specified date/time format, as accepted by time.strftime().
- style - If format is specified, use this style for the format string. One of '%', '{' or '$' for printf-style, str.format() or string.Template respectively. Defaults to '%’.

basicConfig (**kwargs) Method

This method is used to config the logging System.

Syntax:-

basicConfig(**kwargs)

- stream – Use the specified stream to initialize the StreamHandler. Note that this argument is incompatible with filename - if both are present, a ValueError is raised.
- handlers – If specified, this should be an iterable of already created handlers to add to the root logger. Any handlers which don’t already have a formatter set will be assigned the default formatter created in this function. Note that this argument is incompatible with filename or stream - if both are present, a ValueError is raised.
- force - If this keyword argument is specified as true, any existing handlers attached to the root logger are removed and closed, before carrying out the configuration as specified by the other arguments.

Levels

Level	Numeric Value
NOTSET	0
DEBUG	10
INFO	20
WARNING	30
ERROR	40
CRITICAL	50

Methods

- getLogger() – This method returns a logger with the specified name or, if name is None, return a logger which is the root logger of the hierarchy. If specified, the name is typically a dot-separated hierarchical name like ‘a’, ‘a.b’ or ‘a.b.c.d’.

- info(msg) - This will log a message with level INFO on this logger.

- warning(msg) - This will log a message with level WARNING on this logger.

- error(msg) - This will log a message with level ERROR on this logger.

- critical(msg) - This will log a message with level CRITICAL on this logger.

- exception(msg) - This will log a message with level ERROR on this logger.

Format

Format can take a string with LogRecord attributes in any arrangement you like.

asctime – Human-readable time when the LogRecord was created. By default this is of the form ‘2003-07-08 16:49:45,896’ (the numbers after the comma are millisecond portion of the time).

Ex:- %(asctime)s

created – Time when the LogRecord was created (as returned by time.time()).

Ex:- %(created)f

filename – Filename portion of pathname.

Ex:- %(filename)s

LogRecord Attributes

levelname – Text logging level for the message ('DEBUG', 'INFO', 'WARNING', 'ERROR', 'CRITICAL').

Ex:- %(levelname)s

levelno – Numeric logging level for the message (DEBUG, INFO, WARNING, ERROR, CRITICAL).

Ex:- %(levelno)s

lineno – Source line number where the logging call was issued (if available).

Ex:- %(lineno)d

LogRecord Attributes

message – The logged message, computed as msg % args. This is set when Formatter.format() is invoked.

Ex:- %(message)s

name – Name of the logger used to log the call.

Ex:- %(name)s

pathname – Full pathname of the source file where the logging call was issued (if available).

Ex:- %(pathname)s

LogRecord Attributes

args

exc_info

funcname

module

msecs

msg

process

processname

relativecreated

stack_info

thread

threadname