

Digital Career Institute

Python Course - Django Advanced Features



Logging in Django

The logging Python Module

Django uses the Python **logging** module and provides some additional features.



<https://docs.python.org/3/library/logging.html>

Logs, Messages & Levels

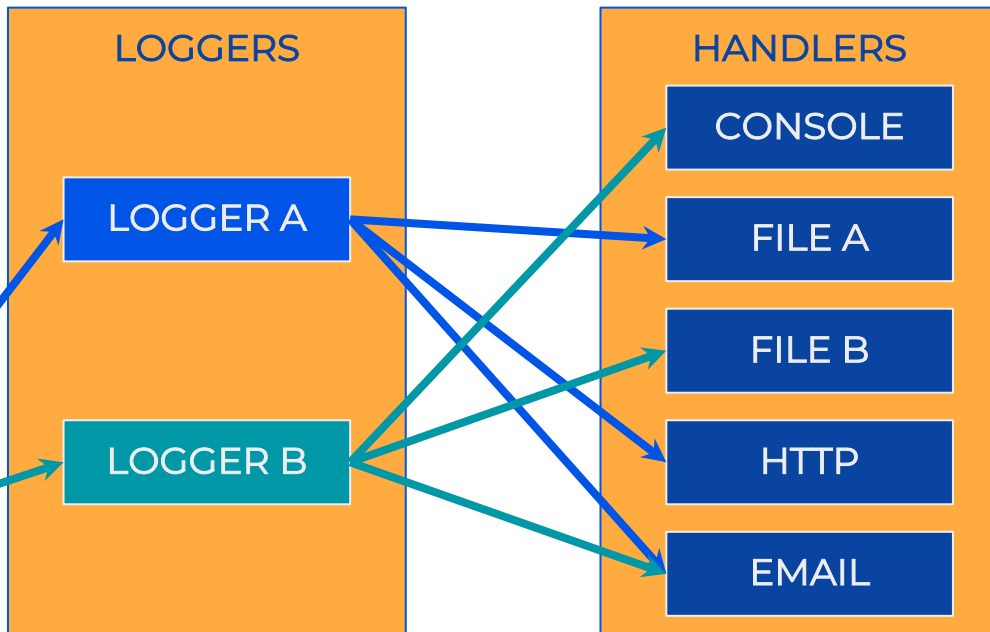
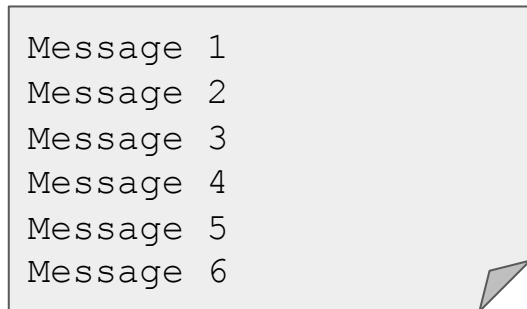
A **log** is a sequence of **messages** that can be segmented into different levels of severity, called **log levels** or **logging levels**.

LOGGING LEVELS	
DEBUG	Low level system information for debugging purposes.
INFO	General system information.
WARNING	Information describing a minor problem that has occurred.
ERROR	Information describing a major problem that has occurred.
CRITICAL	Information describing a critical problem that has occurred.

Loggers & Handlers

A **logger** is the entry point into the logging system. A **handler** is the component that sends the message somewhere.

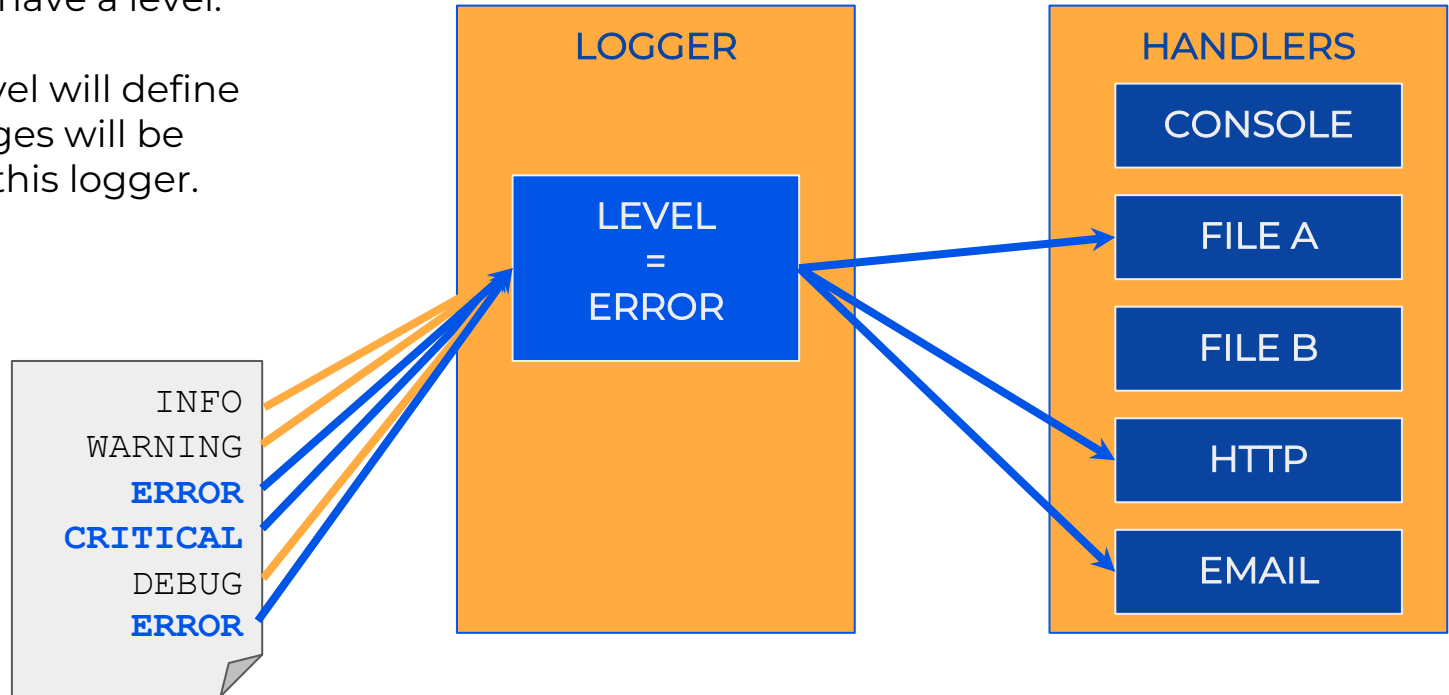
The logger is a stream of input messages. The handler is a stream of output messages.



Loggers

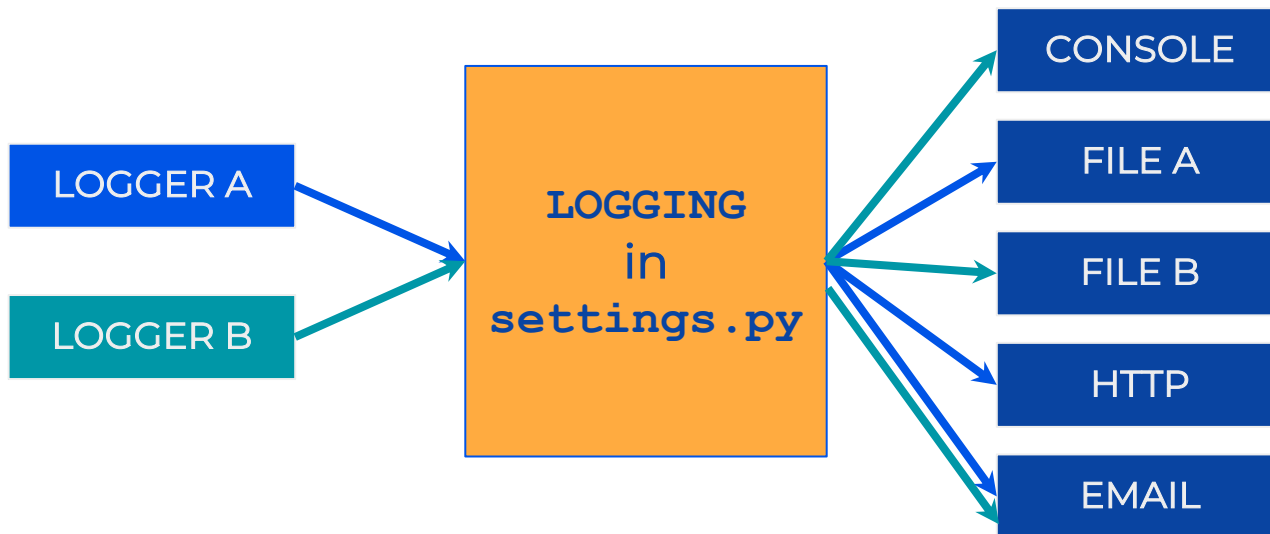
Loggers may have a level.

The logger level will define which messages will be managed by this logger.



Loggers, Handlers & Settings


The constant **LOGGING** in **settings.py** works a bit like the URLconf in the URL dispatcher, it can be used to identify which **handlers** should each **logger** use.



Settings: LOGGING

hello/settings.py

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'console': {
            'class': 'logging.StreamHandler',
        },
    },
    'root': {
        'handlers': ['console'],
        'level': 'INFO',
    },
}
```



Default for all loggers

Creating and Using Loggers in Django

shop/views.py

```
import logging

logger = logging.getLogger("my_logger")

def home(request, key=None):
    logger.info("A user has visited the home page.")
    logger.debug(request)
    logger.warning("The page is not public, yet.")
    if not key:
        logger.error("No key has been provided")
    if key == "I'm g0nn4 h4ck U":
        logger.critical("The site is under attack!")
```

A new logger can be obtained with the **getLogger** method.

logger has a method for each logging level.

Creating and Using Loggers in Django

```
Django version 3.2.6, using settings 'hello.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
A user has visited the home page.
The page is not public, yet.
No key has been provided
```

The `logger.debug(request)` line from the previous slide does not show here, because according to the previous `settings.py`, the logger level is INFO and the DEBUG level is lower than the INFO level.

Settings: Loggers

hello/settings.py

```
LOGGING = {  
    ...  
    'root': {  
        'handlers': ['console'],  
        'level': 'INFO',  
    },  
    'loggers': {  
        'my_logger': {  
            'handlers': ['console', 'file_a'],  
            'level': 'WARNING',  
        }  
    }  
}
```

Default for all loggers

Specific for the logger named my_logger

Different and multiple handlers may be defined for specific loggers by their name.

Settings: Handlers

hello/settings.py

```
LOGGING = {
    ...
    'handlers': {
        'console': {
            'class': 'logging.StreamHandler',
        },
        'file_a': {
            'level': 'WARNING',
            'class': 'logging.FileHandler',
            'filename': '/path/to/file.log'
        },
    },
}
```

All levels will be sent to the console.

Only warnings and above (errors and critical errors) will be stored in a file.

HANDLERS

From the logging module:

- StreamHandler
- FileHandler
- SMTPHandler
- HTTPHandler
- ...

From Django:

- AdminEmailHandler

We have seen how to handle each log message based on its log level.

Filters are used to define how to handle each log message based on other premises different than the log level.

hello/settings.py

```
DEBUG = True
LOGGING = {
    ...
    'filters': {
        'require_debug_true': {
            '()': 'django.utils.log.RequireDebugTrue',
        },
    },
}
```

Logging Filters

hello/settings.py

```
LOGGING = {
    ...
    'loggers': {
        'my_logger': {
            'handlers': ['console', 'file_a'],
            'filters': ['require_debug_true']
        }
    }
    'handlers': {
        'console': {
            'class': 'logging.StreamHandler',
            'filters': ['require_debug_true']
        },
    }
}
```

Filters can be applied to both **handlers** and **loggers**.

Logging Formatters

Formatters are used to define which data should be stored and how should it be presented.

hello/settings.py

```
LOGGING = {
    ...
    'formatters': {
        'verbose': {
            'format': '{levelname} {asctime} {module}
{process:d} {thread:d} {message}',
        },
    },
}
```

The variables that can be used are those of the logging Python module:
<https://docs.python.org/3/library/logging.html#logrecord-attributes>

Logging Formatters

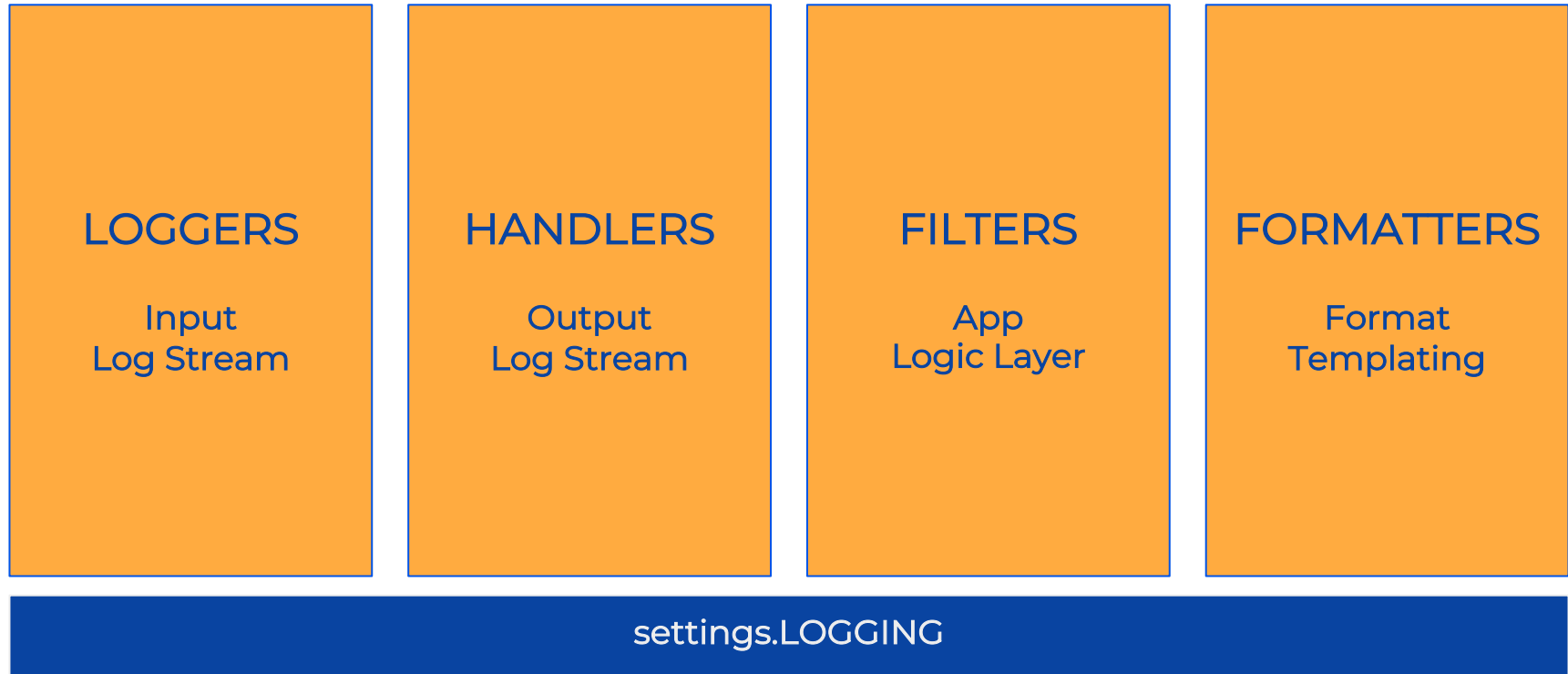
hello/settings.py

```
LOGGING = {  
    ...  
    'handlers': {  
        'console': {  
            'class': 'logging.StreamHandler',  
            'formatter': 'verbose'  
        },  
    },  
}
```

Formatters only apply to
handlers.

```
WARNING 2021-09-20 19:31:17,062 views 1790 140388092057344 The page is not  
public, yet.  
ERROR 2021-09-20 19:31:17,062 views 1790 140388092057344 No key has been  
provided
```


Logging Summary



We learned ...

- The difference between loggers, handlers, filters and formatters.
- That loggers use handlers to “do something” with the log messages.
- That Django uses the Python module **logging** and how to log messages with Django.
- That the **settings.LOGGING** constant defines our custom configuration for using loggers, handlers, filters and formatters.

Documentation

Sessions

- <https://docs.djangoproject.com/en/4.2/topics/http/sessions/>

Testing & Logging

- <https://docs.djangoproject.com/en/4.2/topics/testing/>
- <https://docs.python.org/3/library/unittest.html>
- <https://docs.djangoproject.com/en/4.2/topics/logging/>
- <https://docs.python.org/3/library/logging.html>

Middleware

- <https://docs.djangoproject.com/en/4.2/topics/http/middleware/>
- <https://docs.djangoproject.com/en/4.2/ref/middleware/>



THANK YOU

Contact Details
DCI Digital Career Institute gGmbH