Logging in Flask applications

Abdul Rehman
MACHINE LEARNING ENGINEER

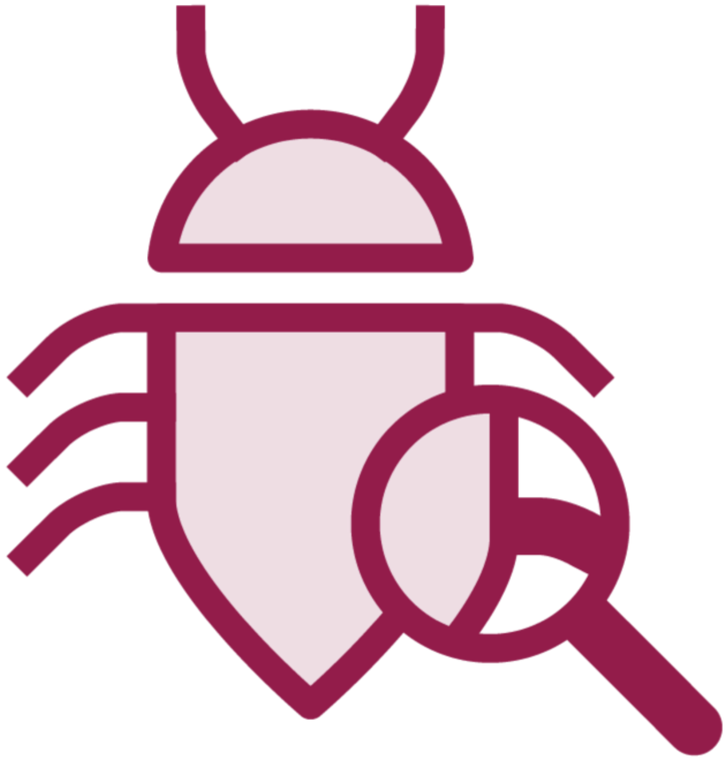
@aPythonist www.pythonist.org

Overview

- Why logging is important?
- Implementation options
- Setup flask-debugtoolbar
- Basics of logging in Python
- Configure custom log handlers
- Implement logging in BookLi app

Let’s get started!

Flask-debugtoolbar



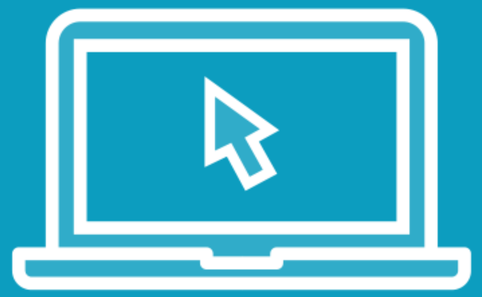
A Flask extension

Aids development by adding debugging

Gives information like

- Bottlenecks in view rendering code
- Values in HTTP Headers

Demo

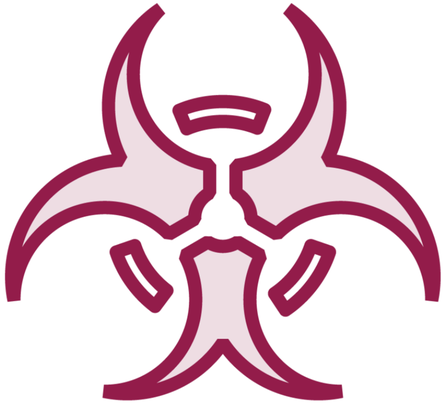


Use Flask-debugtoolbar in BookLi

Logging in Flask

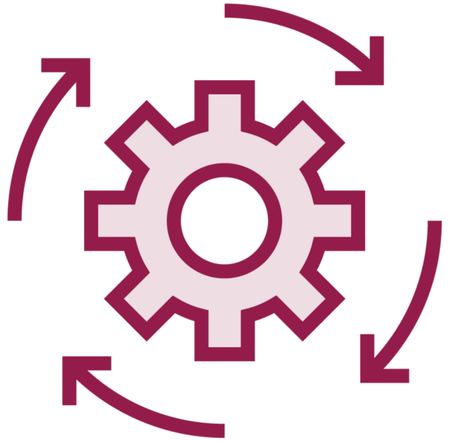
What

What logging means?



Why

Why logging important?



How

Logging in Python

A very useful tool

Understanding of the flow

Discover scenarios

Your extra eyes

Performance analysis

Easy decision making

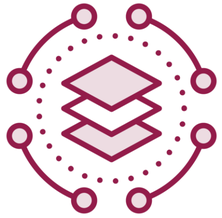
Logging in Flask - Example

Running a Python script you want:

- What parts executed?
- At what time?
- What values variables holds?

You may just use “print”!

On larger projects you want more structured messages.



Code could go through different stages like Testing etc.

If-else and print can make this time consuming.

Python Logging module

- Included in standard library
- Log message gives much info
- When & Where log fires
- Log context like prcoess
- Procedure list
- Talking points

Log Levels

Debug

Used to give Detailed information: 10

Info

Used to Confirm that things are working: 20

Warning

Used as indication for unexpected things: 30

Error

Tells about a serious problem in app: 40

Critical

Tells about serious error which stop the app: 50

By default, a new logger has the NOTSET level, and as the root logger has a WARN level. It's numeric value is 0

Logging Formatting

Formatting Enriches a log message by adding context information to it

It's useful to know when the log is sent, where and additional context such as the thread

Logging Formatting

Formatting.py

```
"%(asctime)s - %(name)s - %(levelname)s - %(funcName)s:%(lineno)d - %(message)s"
```

```
2018-02-07 19:47:41,864 - a.b.c - WARNING - <module>:1 - hello world
```

Logging Handler

The log handler is the component that effectively writes or displays a log.

StreamHandler

Display the log messages in the console.

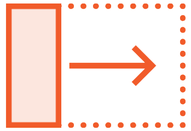
FileHandler

Write the log messages in a file.

SMTPHandler

Sending logs to your email address.

Each Log Handler has:



A formatter which adds context information to a log.

A log level that filters out logs whose levels are inferior.

Logging in Python

Getting started

How to get started with logging quickly

Advanced features

Introduction to some advanced features

Python Logging Module

Ready-to-use

Powerful

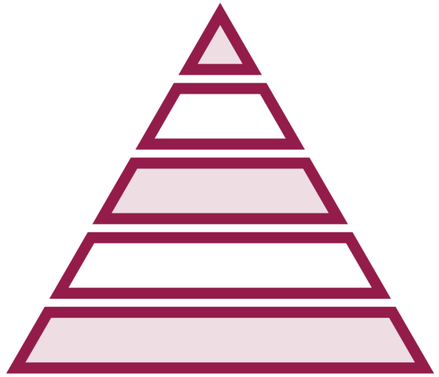
Satisfy the needs of beginners & enterprise

Used by third-party libraries

Demo

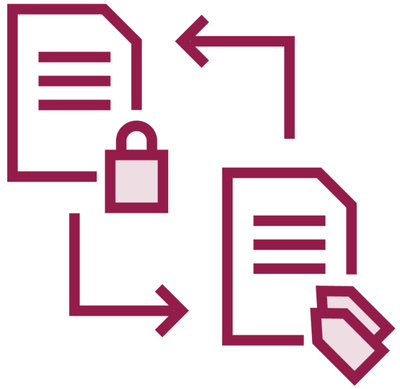
Use logging Module

Configure the Logger: basicConfig()

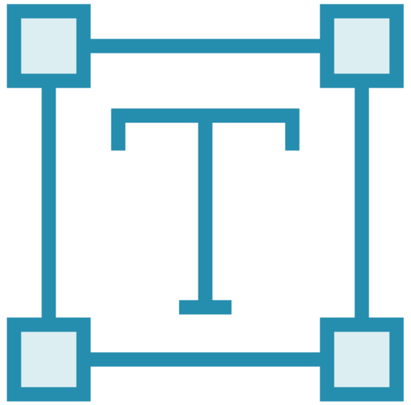


Level

Filename

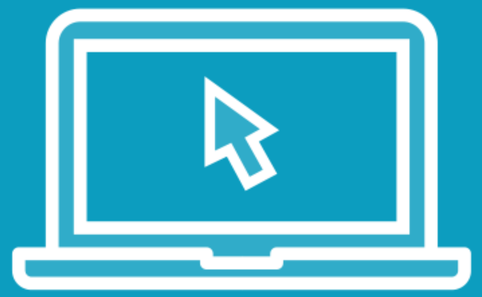


Filemode



Format

Demo



Use basicConfig() in BookLi

Filemode **“w”** will overwrite the existing logs and default filemode is **“a”** which means append.

basicConfig()

Even further customizable:

- Style
- Handlers
- Stream

basicConfig() can only be called once in your application

Functions debug, info, warning, error and critical also call basicConfig automatically without args

Log Formatting

String variable

Can pass any variable represented as a string from your program as log message

LogRecord elements

You can add any element from LogRecord list easily like Process_ID etc

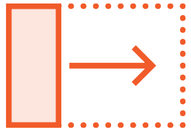
Logging Formatting

Formatting.py

```
logging.basicConfig(format='%(process)d - %(name)s - %(message)s')

logging.warning('This is a Warning')
```

```
15653 - WARNING - This is a Warning
```

Logging Variable Data



Add dynamic information in the logs

Commonly, we format the string with a variable in separate lines

It can be formatted on single line using f-string

Logging Variable Data

VariableData.code

```
import logging
if logged_in() == False:
    err = 'Login'
else:
    err = 'Not Login'
logging.error(f'{err} raised an error.')
```

Capturing Stack Traces

Full Stack Trace

Exception – exc_info

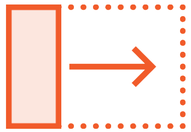
Capturing Stack Traces

StackTrace.py

```
try:

    c = a / b

except Exception as e:

    logging.error("Exception
            occurred", exc_info=True)
```

```
ERROR:root:Exception occurred
Traceback (most recent call last):
File "exceptions.py", line 6, in <module>
c = a / b
ZeroDivisionError: division by zero
```

Use the **logging.exception()** method, which logs a message with level **ERROR** and adds exception information to the message



How to define custom log handlers

Add custom log handlers to your application

How to remove existing custom log handler

We Covered

Log Levels

Default Loggers

Basic Configuration

Logging in Flask

Flask use *app.logger* for log messages

You can also use *app.logger* for logs

Default logging mechanism in Flask

Logging Basics

In general

We use ***logging.info*** for an info level log message

In Flask

We can use ***app.logger.info*** for a log message

Code Example

```
app.logger.info('A sample log message')
```

Demo

Default logging in Flask

Customize Logging Configuration in Flask

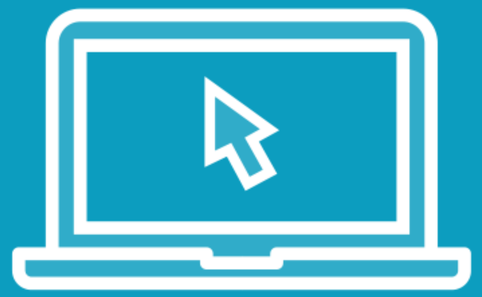
DictConfig

A python dictionary that provides different configuration values

FileConfig

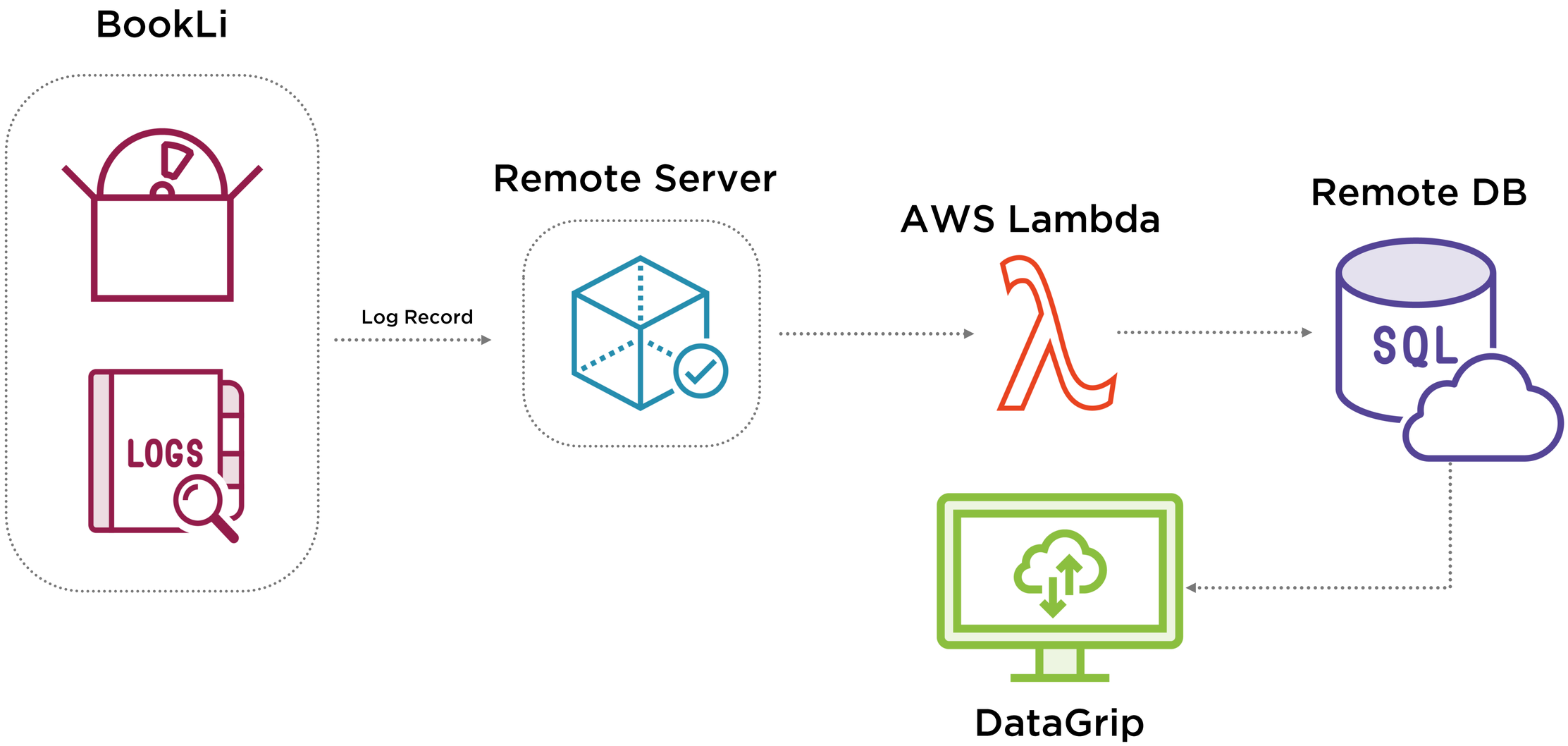
A configuration file for the same purpose as the ***dictConfig***

Demo



Configuration using dictConfig

Workflow of Custom Log Handler



Log Handlers

Dictate how log entries handled

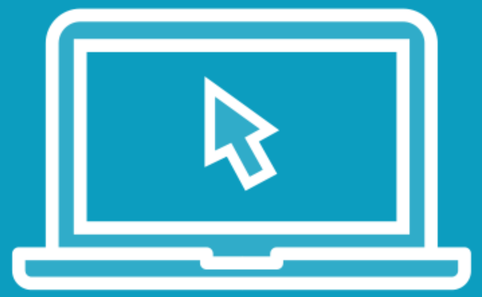
File Handler send logs to file

HTTP Handler send logs to remote server

We can write custom log handlers:

- Subclass from *logging.Handler*
- Define *emite* method
- Follow a structure similar from earlier slide

Demo



Implement a Custom Log Handler in BookLi

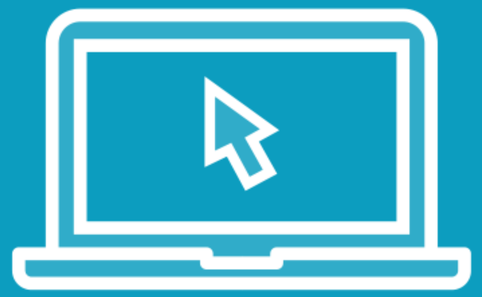
Let’s use our sample-flask-app as the remote server in current scenario

Inject Request Information in Logs

Help us to debug error appropriately

How to achieve that?

Demo



Inject Request URL to logs

Summary

Introduction to Logging & its importance

Python Logging module

Explore Flask-debugtoolbar

Customize the logging module configs

Custom LogHandler & LogFormatter

Add & Remove log handler in Flask

Make Flask applications reliable!