




# Log it

 @guilatrova

 hello@guilatrova.dev

 guilatrova

 guicommits.com

Pitch



Log it,  
but please,  
do it RIGHT



Look it  
but please,  
do it RIGHT

```
print("Something bad happened")
```



# Log it

```
print("Something had hannened")
```

```
logger.error(f"Malformed data returned from API: {str(exception)}")
```

# do it RIGHT



Look it



```
print("Something had hannened")
```



```
logger.error(f"Malformed data returned from API: {str(exception)}")
```



```
logger.info("Listening...")
```

do it RIGHT



```
logger.error("API error. Likely is not a valid connection")
```



```
logger.info("Listening...")
```

```
print("Something had hannened")
```



```
logger.error(f"Malformed data returned from API: {str(exception)}")
```

do it RIGHT



```
logger.error("API error. Likely is not a valid connection")
```



```
logger.info("Listening...")
```

```
print("Something had happened")
```



```
logger.error(f"Malformed data returned from API: {str(exception)}")
```



```
logger.error("Operation failed due to an internal error")
```

# RIGHT





# SO MUCH NOISE





# Who I am

I'm him 🙌

- Formal Education: Information Systems
- 8 years coding
- A generalist (python, devops, front)
- \*
- Currently making software @ Lumos Identity

## Important Note:

I swear I'm way more handsome in person.



# Gui Latrova



# Who I am

- ↳ ~~Formal Education: Information Systems~~
- 8 years coding
  - A generalist (python, devops, front)
  - \*
  - Currently making software @ Lumos Identity



Gui Latrova



# Who I am

- ~~Formal Education: Information Systems~~
- 8 years coding
- A generalist (python, devops, front)
- ~~My designs suck~~
- Currently making software @ Lumos Identity



Gui Latrova



# ROADMAP

You're here



## Problem

We're going to introduce an example using different and complex **integrations** with GitHub, AWS, and Slack.

## Definition

What is a log?  
Actually, **what** to log?  
**when** to log?

## Applying

Let's add logs to our **examples**.  
There's way more than **info** and **error**, and I can prove!

## Setup

Let's break down how to **setup the logging lib** (finally some code!)



```
logger.info(  
    "Lets get started"  
)
```

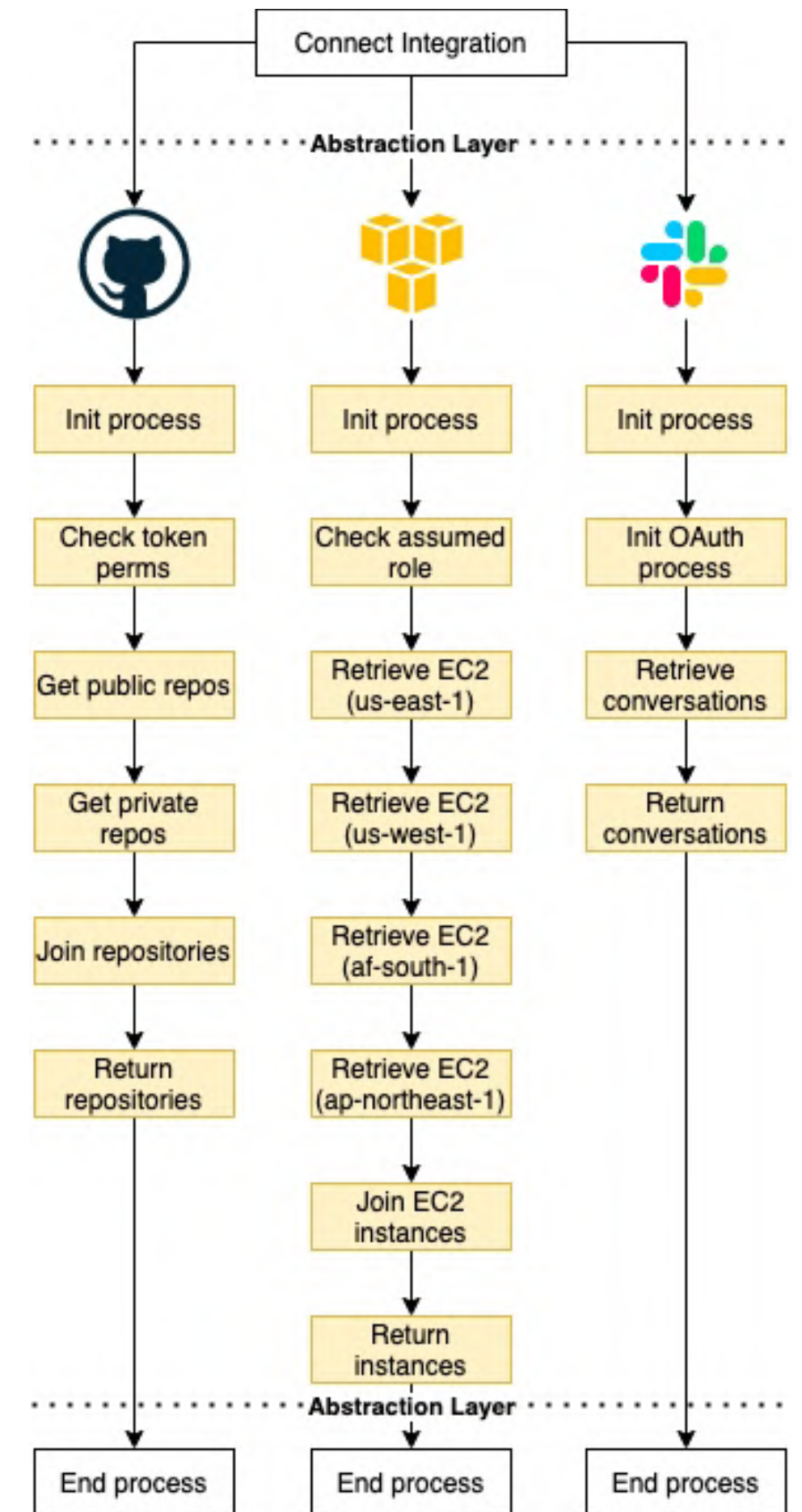
👉 code linted with black

# Problem

- Users can **connect multiple integrations**
- Resources are **unique** based on the integration
- Each integration is **uniquely complex**
- Each integration may produce different **errors** at **any** time



More common on Fridays, 🖐️  
when you want to leave early





Definition

Clearly not this 🙅 ☐

It's a piece of information that is

- Contextual
- Reactive
- Descriptive



## Definition

Clearly not this 🙅 ☐

# It's a piece of information that is

- Contextual
- Reactive
- Descriptive

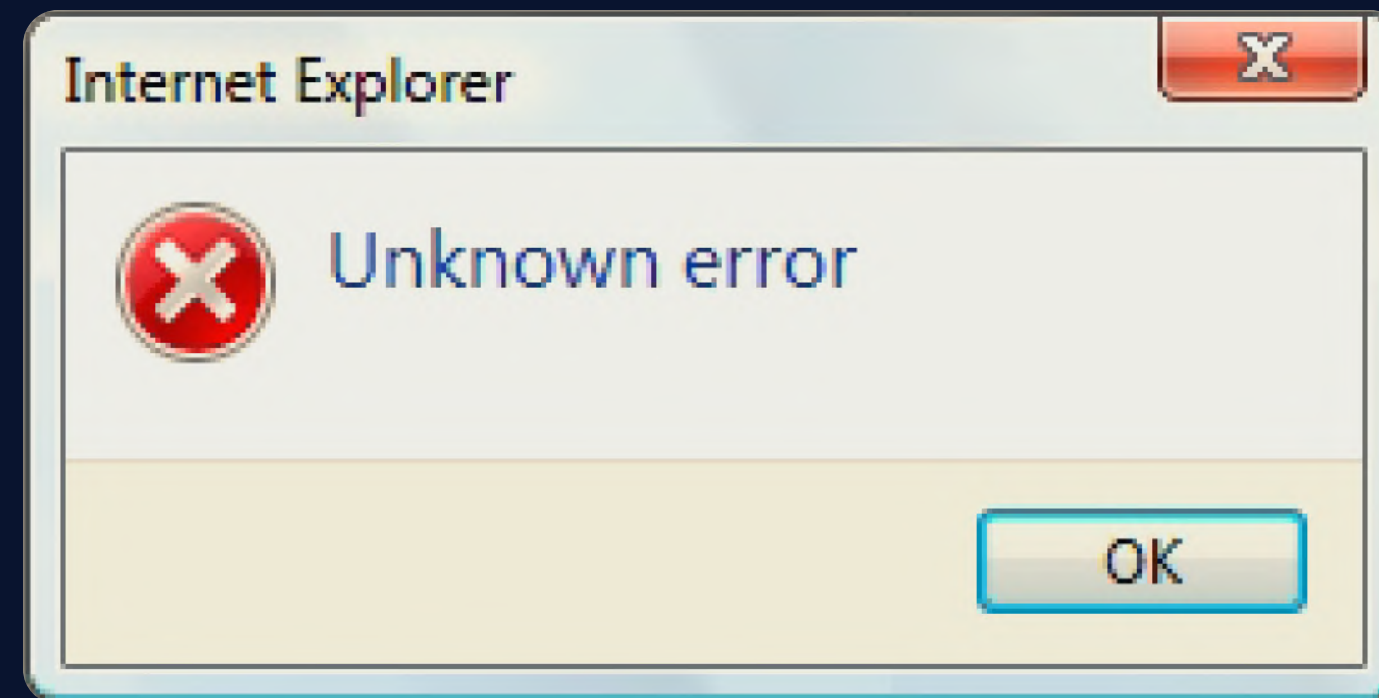
```
logger.error("Operation connect failed")
```

Definition

It's always descriptive then?

Definition

It's always descriptive then?

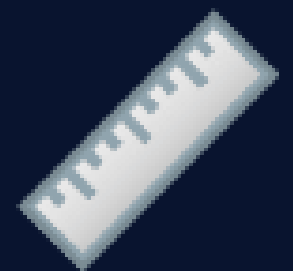


Definition

# When to log?

Definition

# When to log?



Definition

# When to log?

“Logs should tell you a **story**, every story has a **beginning**, middle, and **end**”

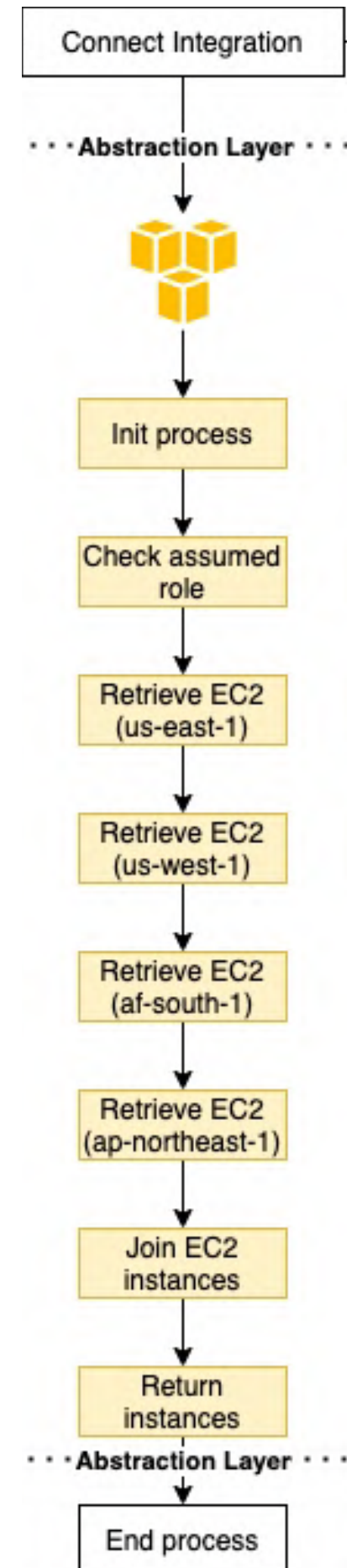


Applying

# How a **good logs** looks like?

Success scenario

Message	What I know?	Context
Connecting to AWS	integration AWS operation started	<u>Log attributes</u> should allow me to find out who triggered it
Retrieved instances from all regions	One relevant progress has been made	Same above
Connection to AWS has been successful	AWS operation finished	Same above



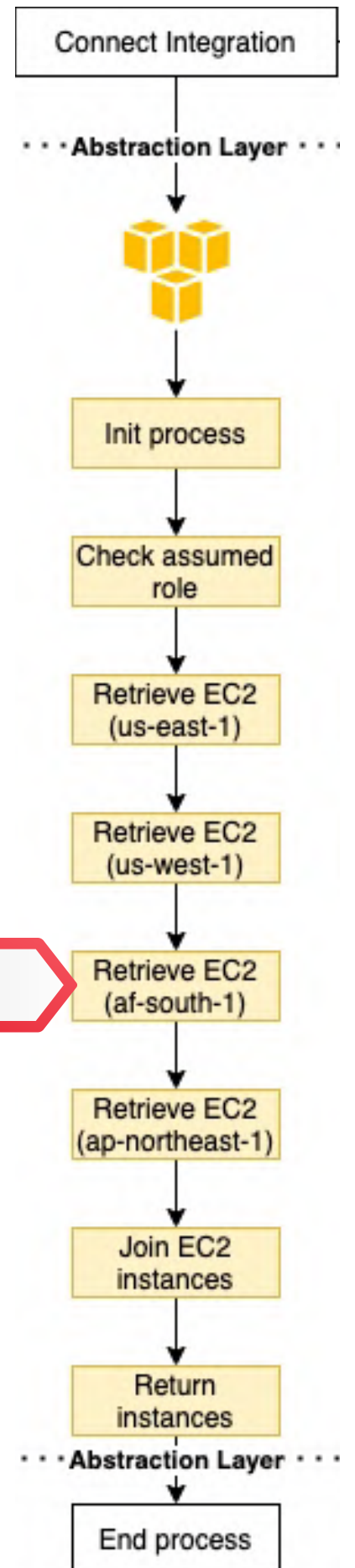


Applying

# How a **good logs** looks like?

Failed scenario

Message	What I know?	Context
Connecting to AWS	integration AWS operation started	<u>Log attributes</u> should allow me to find out who triggered it

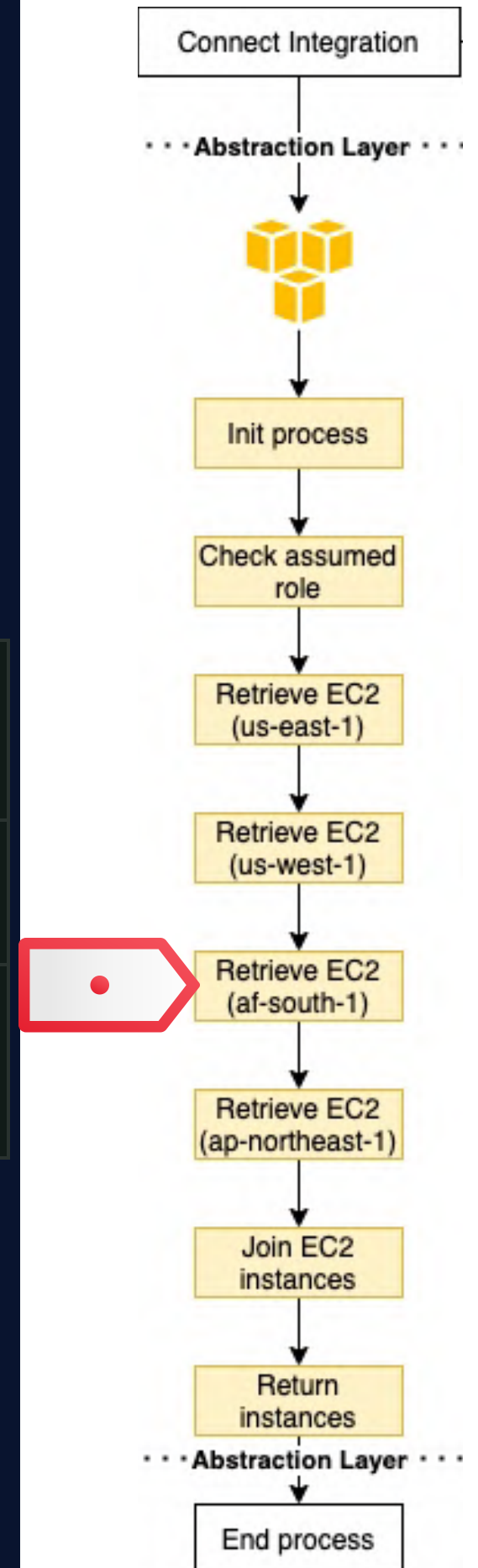


Applying

# How a **good logs** looks like?

Failed scenario

Message	What I know?	Context
Connecting to AWS	integration AWS operation started	<u>Log attributes</u> should allow me to find out who triggered it
Failed to retrieve instances from regions af-south-1 when connecting to AWS for user X	<ul style="list-style-type: none"><li>• AWS operation didn't finish</li><li>• region af-south-1 failed</li><li>• user X got affected</li></ul>	I should be able to see the error's stack trace to dive into the "why" it failed



Applying



# Here's a **hack** for you

When writing logs, **ask yourself:**

Applying



# Here's a **hack** for you

When writing logs, **ask yourself:**

“If I read this in the future,  
what I'd wish to understand  
after reading it?”

Applying

# Python Context

Can be added like:



```
logger.info("Connecting to AWS", extra={"user": "X"})
```

# Applying Python Context

Can be added like:



```
logger.info("Connecting to AWS", extra={"user": "X"})
```



```
logger.exception("...")
```

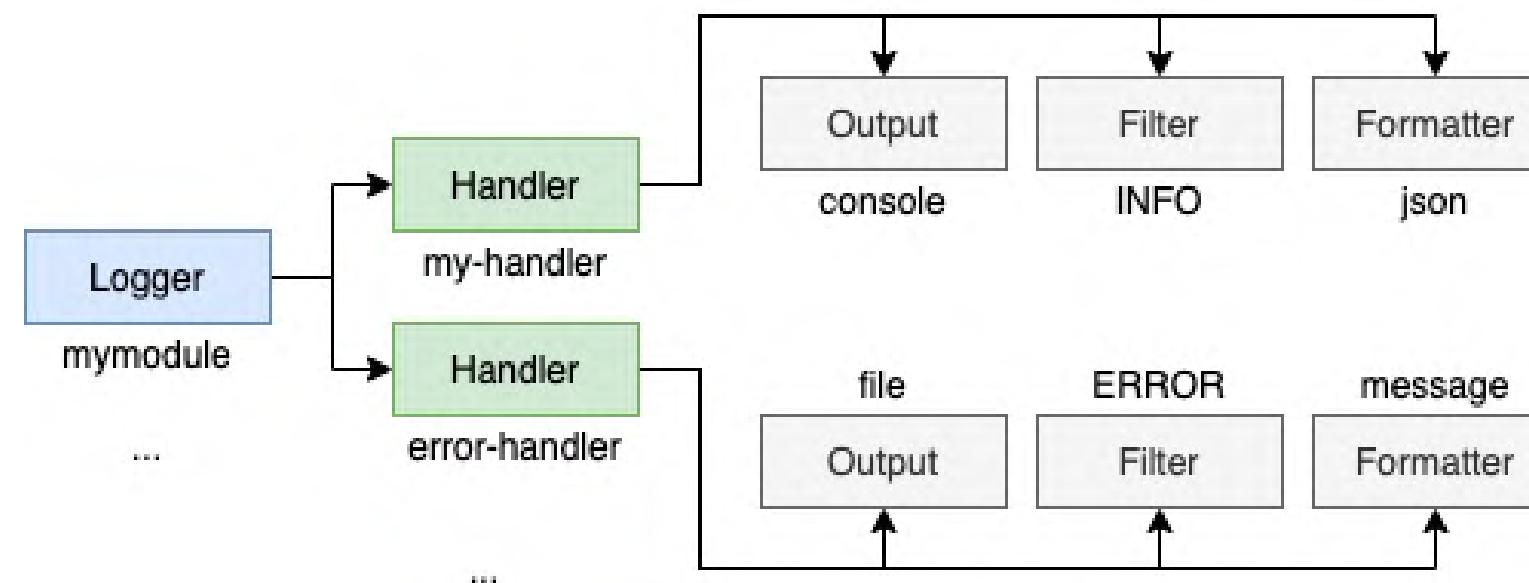
Setup

# logging library



Setup

# logging library



Setup

# logging library

Loggers

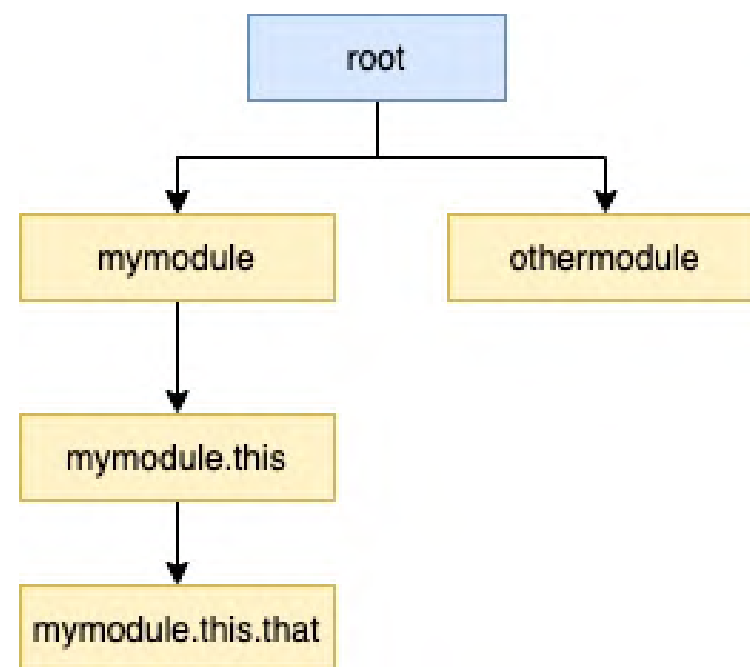


```
import logging  
  
logger = logging.getLogger("any.name.that.i.want")
```

Setup

# logging library

Loggers



```
import logging
```

```
logger = logging.getLogger(__name__)
```

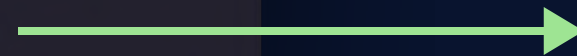
Setup

# logging library

Formatters



```
import logging  
  
logger = logging.getLogger(__name__)  
logger.info("This happened")
```



\* Sharing **lib** at the end of presentation

Setup

# logging library

Filters

```
import logging

logger = logging.getLogger(__name__)

logger.debug("This is not visible")
logger.info("This is visible")
```

```
# Extracted from Python docs
class ContextFilter(logging.Filter):
    USERS = ['jim', 'fred', 'sheila']
    IPS = ['123.231.231.123', '127.0.0.1', '192.168.0.1']

    def filter(self, record): # Add random values to the log record
        record.ip = choice(ContextFilter.IPS)
        record.user = choice(ContextFilter.USERS)
        return True
```

Setup

# logging library

Handlers

Setup

# logging library

dictConfig

```
import logging.config

LOGGING_CONFIG = {
    "version": 1,
    "disable_existing_loggers": False,
    "formatters": { },
    "handlers": { },
    "loggers": { },
    "root": { },
}

logging.config.dictConfig(LOGGING_CONFIG)
```



Setup

# logging library

dictConfig

```
import logging.config

LOGGING_CONFIG = {
    ...,
    "formatters": {
        "default": {
            "format": "%(asctime)s:%(name)s:%(process)d:%(lineno)d " "%(levelname)s %(message)s",
            "datefmt": "%Y-%m-%d %H:%M:%S",
        },
        "simple": {
            "format": "%(message)s",
        },
        "json": {
            "()": "pythonjsonlogger.jsonlogger.JsonFormatter",
            "format": """
                asctime: %(asctime)s
                created: %(created)f
                filename: %(filename)s
                funcName: %(funcName)s
                levelname: %(levelname)s
                levelno: %(levelno)s
                lineno: %(lineno)d
                message: %(message)s
                module: %(module)s
                name: %(name)s
            """
        }
    }
}
```

Setup

# logging library

dictConfig

```
ERROR_LOG_FILENAME = ".tryceratops-errors.log"

LOGGING_CONFIG = {
    ...,
    "formatters": {
        "default": { ... },
        "simple": { ... },
        "json": { ... },
    },
    ...
}
```

```
ERROR_LOG_FILENAME = ".tryceratops-errors.log"

LOGGING_CONFIG = {
    ...,
    "handlers": {
        "logfile": {
            "formatter": "default",
            "level": "ERROR",
            "class": "logging.handlers.RotatingFileHandler",
            "filename": ERROR_LOG_FILENAME,
            "backupCount": 2,
            "maxBytes": 128,
        },
        "verbose_output": {
            "formatter": "simple",
            "level": "DEBUG",
            "class": "logging.StreamHandler",
            "stream": "ext://sys.stdout",
        },
        "json": {
            "formatter": "json",
            "class": "logging.StreamHandler",
            "stream": "ext://sys.stdout",
        },
    },
    ...
}
```

Setup

# logging library

dictConfig

```
LOGGING_CONFIG = {  
    ...,  
    "formatters": { ... },  
    "handlers": {  
        "logfile": { ... },  
        "verbose_output": { ... },  
        "json": { ... },  
    },  
    ...  
}
```

```
LOGGING_CONFIG = {  
    ...,  
    "loggers": {  
        "tryceratops": {  
            "level": "INFO",  
            "handlers": [  
                "verbose_output",  
            ],  
        },  
    },  
    "root": {  
        "level": "INFO",  
        "handlers": [  
            "logfile",  
            "json",  
        ],  
    },  
}
```

Setup

# logging library

Links



/madzak/python-json-logger



[guicommits.com/how-to-log-in-python-like-a-pro](https://guicommits.com/how-to-log-in-python-like-a-pro)



@guilatrova

👉 Sharing a thread with **the links** I promised and **this presentation!**



Thanks!