PYTHON LOGGING

A MEDITATION ON SILENT FAILURES

WHO IS THIS TALK FOR?

WHY LOGGING?

ERRORS VS. FAILURES

SHOULDN'T I JUST TEST MY CODE?

DEPENDENCY CHANGES:
A HORROR STORY

WHAT IS LOGGING?

ANATOMY OF A PYTHON LOGGER

THREE KEY LOGGER FUNCTIONS

- LEVEL
- HANDLER
- FORMATTER

LOGGING LEVELS

- NOTSET
- DEBUG
- INFO
- WARNING
- ERROR
- CRITICAL

WHERE SHOULD MY LOGS GO?

HANDLER

```
import logging

logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)

handler = logging.FileHandler("info.log")
handler.setLevel(logging.INFO)

logger.addHandler(handler)

logger.info("Helpful logging message, id: {}".format(1))
logger.error("{} error encountered on line {}".format(KeyError, 10))
```

OUTPUT:

```
Helpful logging message, id: 1
<type 'exceptions.KeyError'> error encountered on line 10
```

WHAT SHOULD I LOG?

BASIC FORMATING

```
import logging

...

formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
handler.setFormatter(formatter)

logger.setHandler(handler)

logger.info("Helpful log message, id: {}".format(1))
logger.error("{} error encountered on line {}".format(KeyError, 10))
```

OUTPUT:

```
2016-07-29 16:39:59,313 - __main__ - INFO - Helpful logging message, id: 1'
2016-07-29 17:05:45,290 - __main__ - ERROR - <type 'exceptions.KeyError'> error encountered on line 10
```

ADVANCED FORMATTING

- EXTERNAL LIBRARIES

- SUBCLASSING `logging.Formatter`

CREATING YOUR OWN FORMATTER:

```
class MyFormatter(logging.Formatter):
    def format(self, record):
        my_message = dict(
            name=record.name,
            timestamp=str(datetime.datetime.now()),
            level=record.levelname,
            message=record.msg,
            exc_info=record.exc_INFO,
            pathname=record.pathname,
            lineno=record.lineno,
        )
        return json.dumps(my_message)

logger.info("This is a helpful message.")
```

```
{"pathname": "my_file.py", "name": "__main__", "level": "INFO", "timestamp": "2016-07-28 14:11:50.256488", "message": "This is a helpful message", "exc_info": null, "lineno": 30}
```

LogRecord

AVAILALBE ATTRIBUTES:

name
levelname
levelno
pathname
lineno
msg
args
exc_info
func

HOW MUCH IS TOO MUCH?

RotatingFileHandler

```
import logging

handler = logging.RotatingFileHandler(path, maxBytes=20,
                                      backupCount=5)
```

LIMITS THE MAXIMUM SIZE OF A LOGFILE

OPTIONAL BACKUPS APPENDED WITH `*1.log`, `*2.log` ETC

THIS FILE ROTATES AS IT APPROACHES 20 BYTES AND KEEPS 5 BACKUPS

TimedRotatingFileHandler

```
handler = logging.TimedRotatingFileHandler(path,
                                           when="d",
                                           interval=2,
                                           backupCount=5)
```

LIMITS THE AMOUNT OF TIME A LOG WILL STICK AROUND

AVAILABLE INTERVALS -> SECS, MINS, HOURS, DAYS, OR WEEKDAYS (0-6)

THIS FILE ROTATES EVERY 2 DAYS WITH 5 BACKUP LOG FILES

I HAVE LOGS!

NOW WHAT?

THANK YOU

SLIDES AND PRESENTER NOTES AVAILABLE ON GITHUB

@JLUNREIN | JESSUNREIN.COM