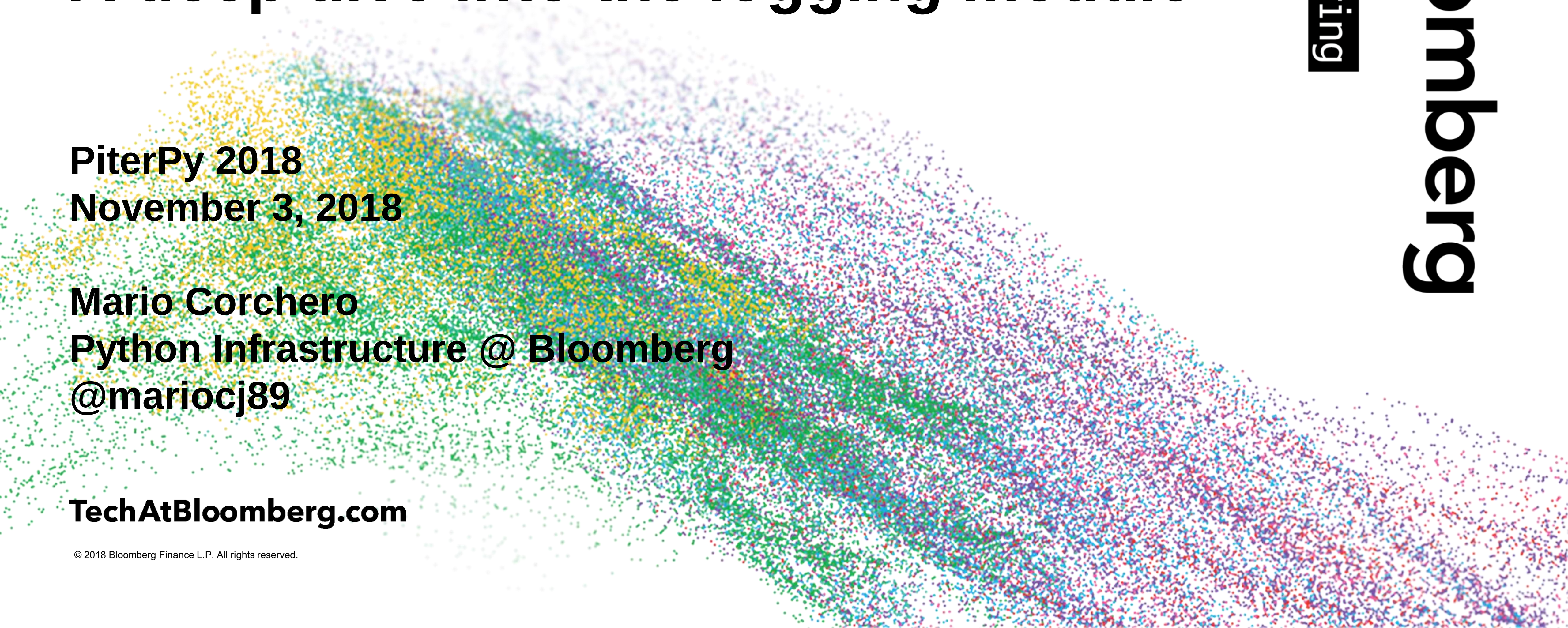
Effortless Logging:

A deep dive into the logging module



PiterPy 2018
November 3, 2018

Mario Corchero
Python Infrastructure @ Bloomberg
@mariocj89

Bloomberg Engineering

TechAtBloomberg.com

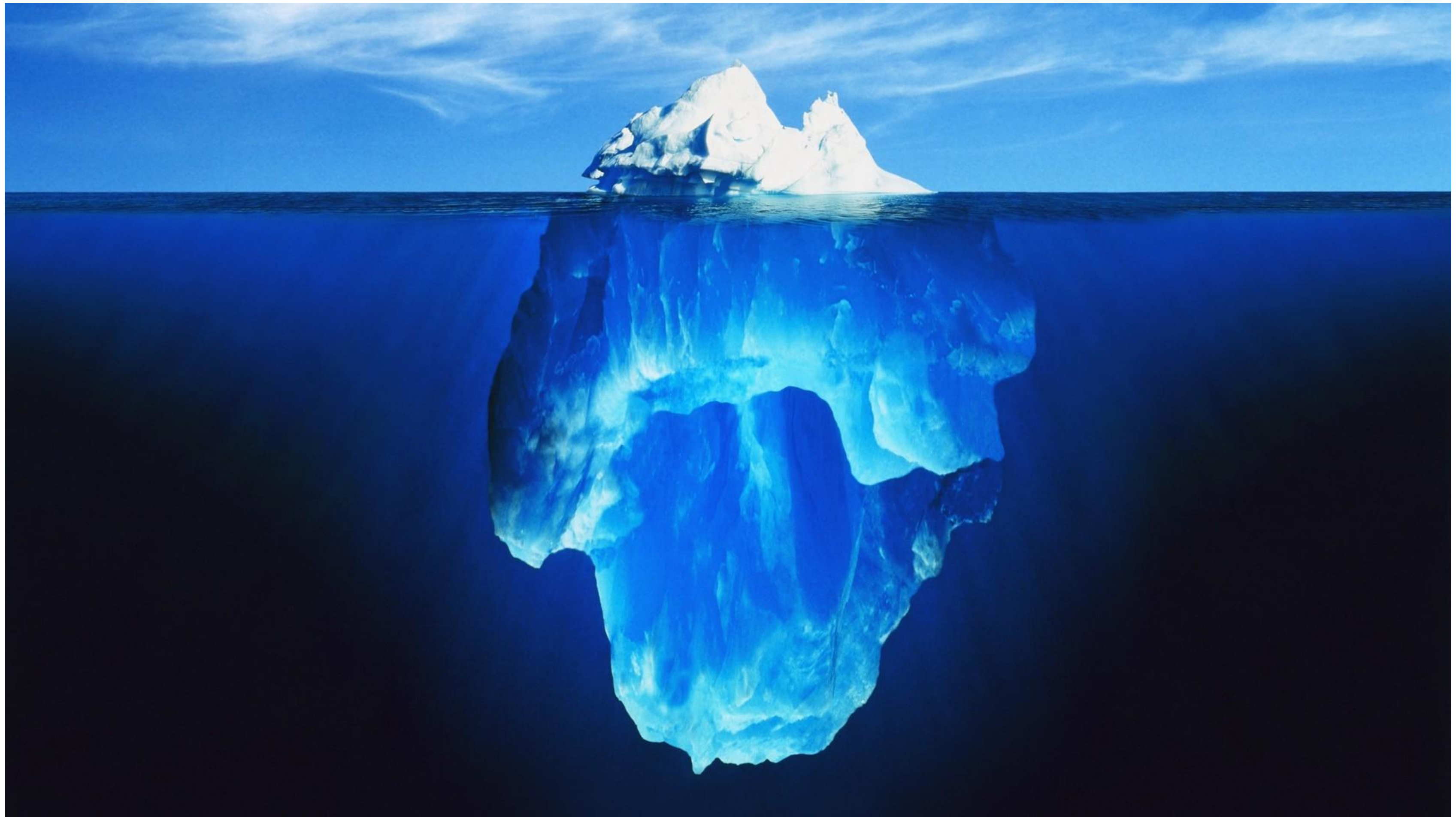
Agenda

- Why logging matters
- How logging works
- How to use it
- How to configure it
- Sailing to the guts of logging
- Sample recipes
- Q&A

Why logging matters

Versatility & Configurability

How logging works



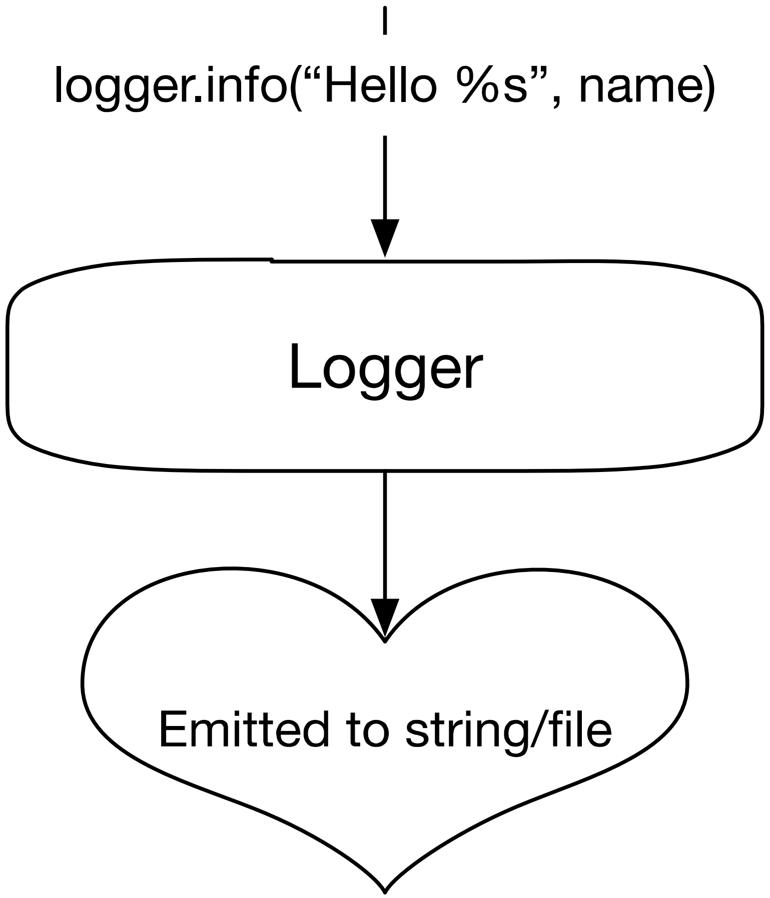
TechAtBloomberg.com

Bloomberg

Engineering

Logger

Logger



TechAtBloomberg.com

Bloomberg

Engineering

Record

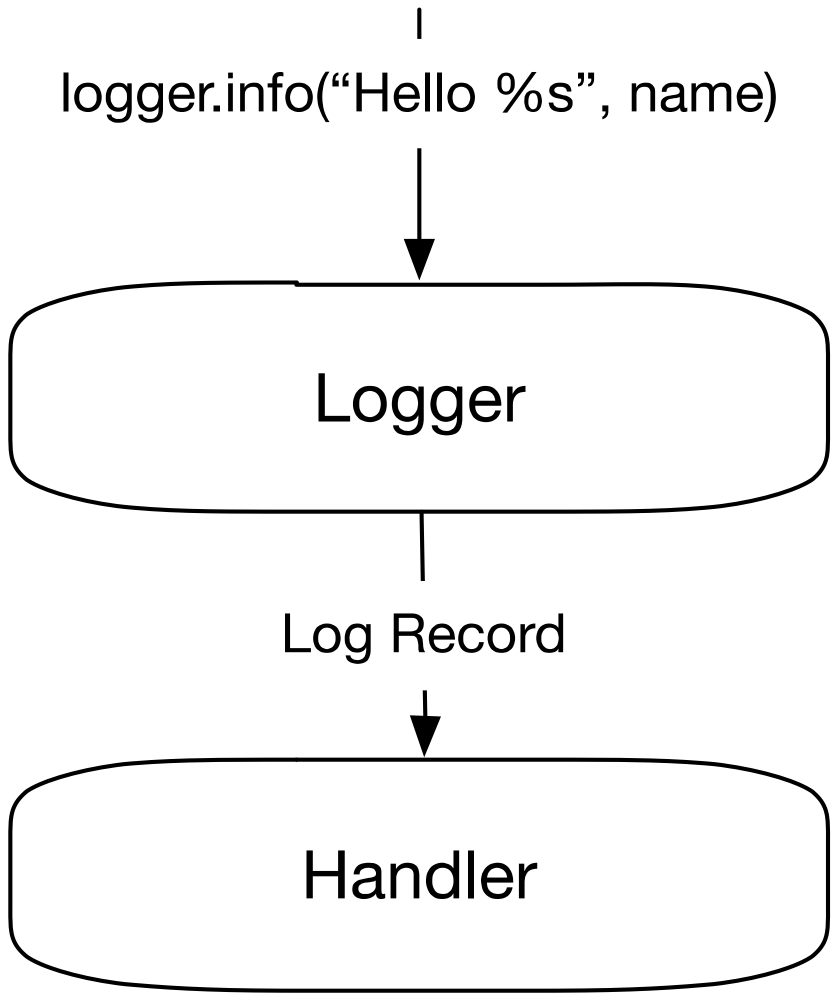
TechAtBloomberg.com

Bloomberg

Engineering

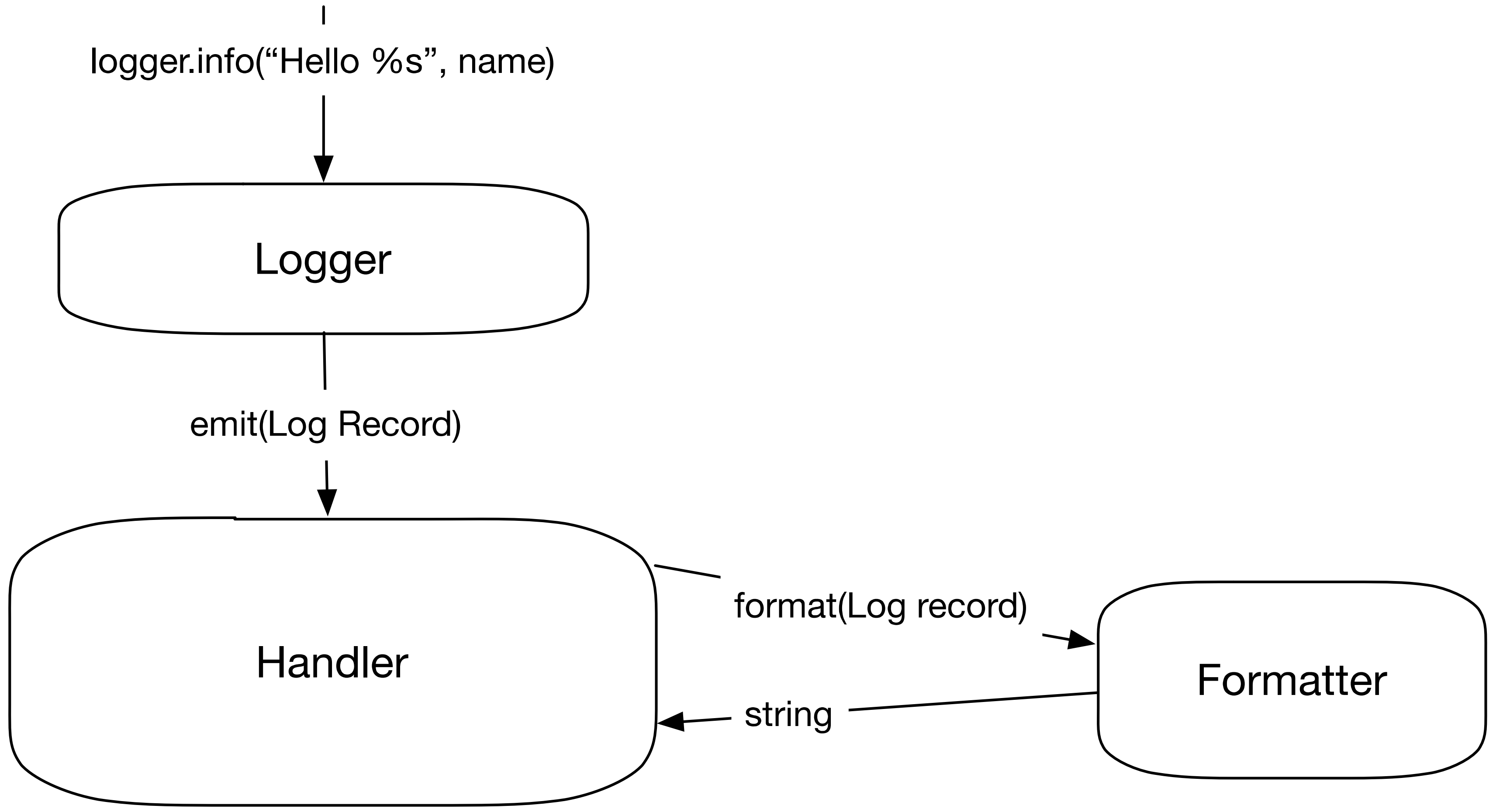
Handler

Handler



Formatter

Formatter



TechAtBloomberg.com

Bloomberg
Engineering

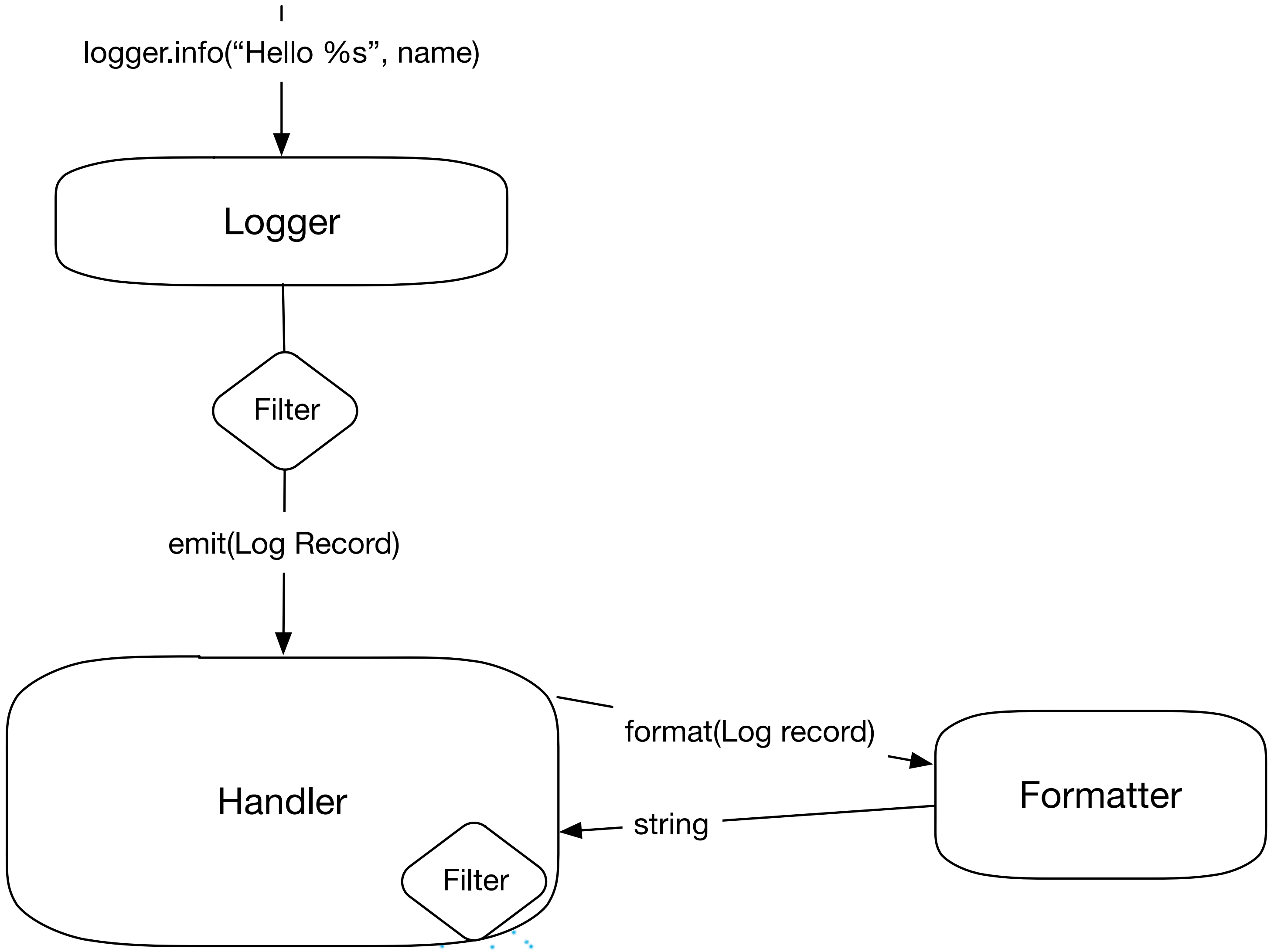
Filter

TechAtBloomberg.com

Bloomberg

Engineering

Filter

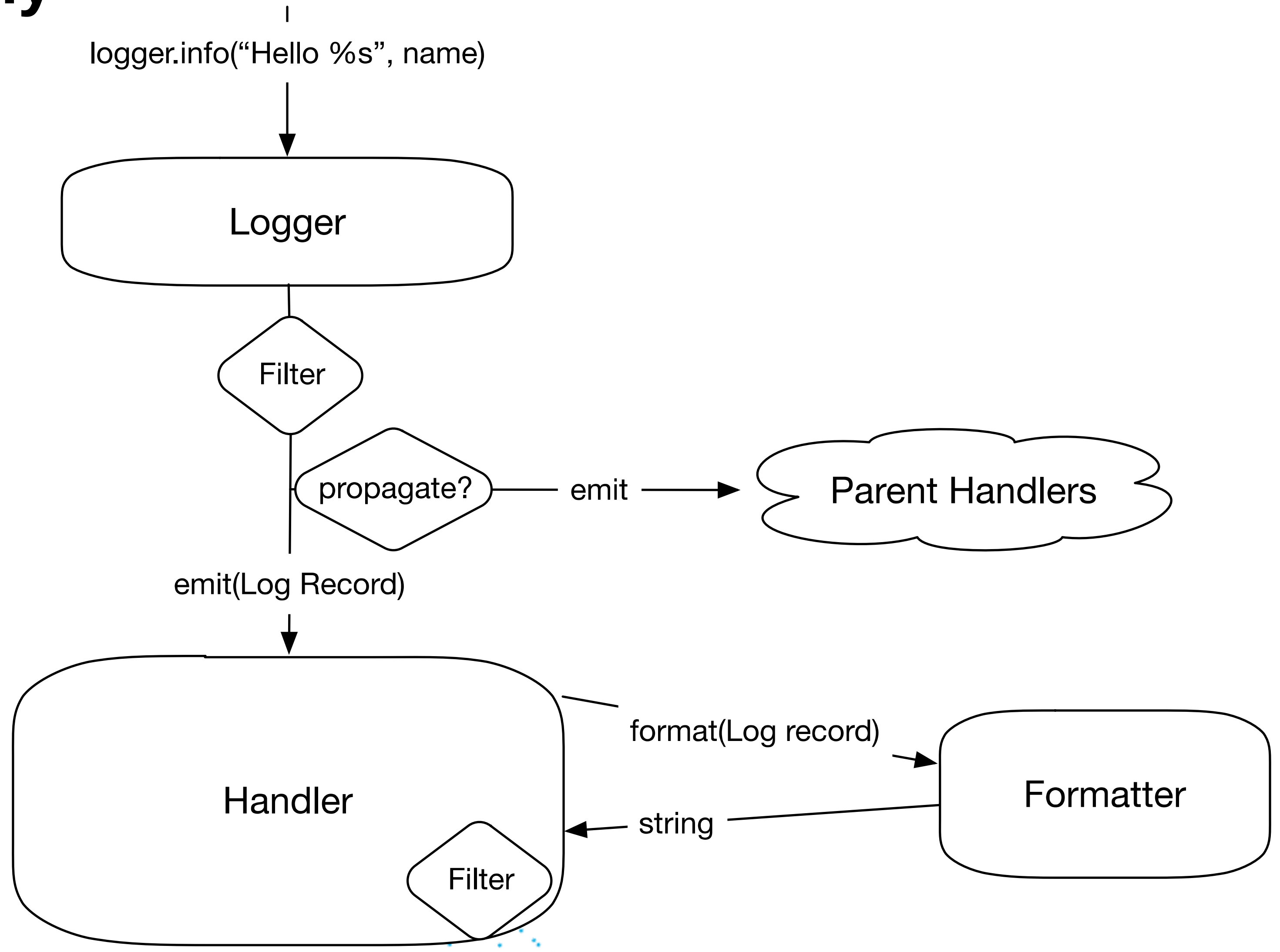


Bloomberg
Engineering

The hierarchy

```
logger = logging.getLogger(”parent.child”)
```

The hierarchy



Bloomberg

Engineering

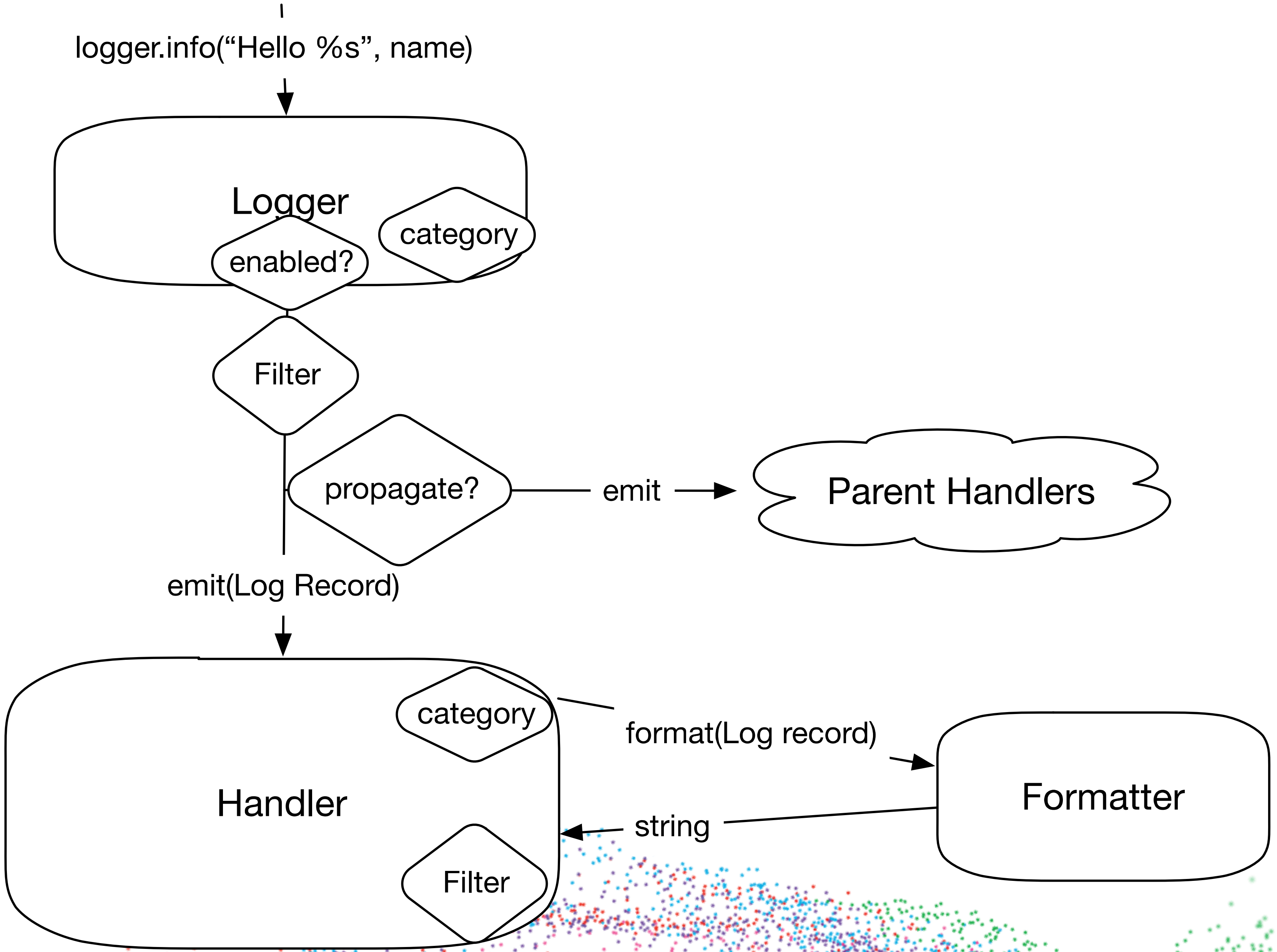
The actual flow

TechAtBloomberg.com

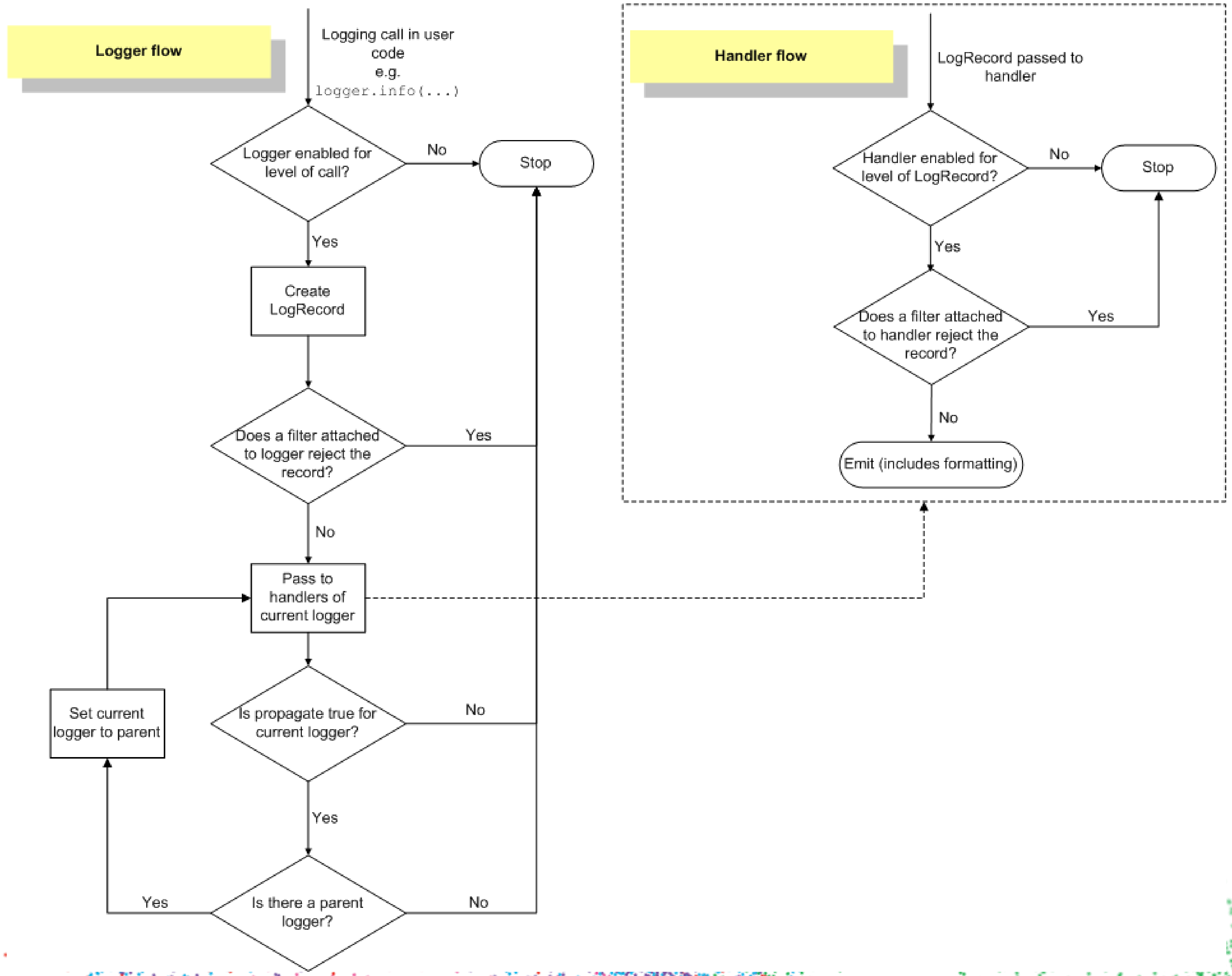
Bloomberg

Engineering

The actual flow



The actual flow


Logger flow
Logging call in user code e.g. logger.info(...)
Logger enabled for level of call?
No
Stop
Yes
Create LogRecord
Does a filter attached to logger reject the record?
Yes
No
Pass to handlers of current logger
Is propagate true for current logger?
No
Yes
Set current logger to parent
Is there a parent logger?
Yes
No
Handler flow
LogRecord passed to handler
Handler enabled for level of LogRecord?
No
Stop
Yes
Does a filter attached to handler reject the record?
Yes
No
Emit (includes formatting)

How to use it

TechAtBloomberg.com

Bloomberg

Engineering

How to use it

```
import logging

def sample_function(secret_parameter):
    logger = logging.getLogger(__name__)
    logger.debug (”Going to perform magic with '%s'”,  secret_parameter)
    ...
    try:
        result = do_magic(secret_parameter)
    except IndexError:
        logger.exception("OMG it happened again, someone please tell Laszlo")
    except Exception:
        logger.info("Unexpected exception", exc_info=True)
        raise
    else:
        logger.info("Magic with '%s' resulted in '%s'", secret_parameter, result, stack_info=True)
```

TechAtBloomberg.com

Bloomberg
Engineering

Common misuses

TechAtBloomberg.com

Bloomberg Engineering

Common misuses

```
except Exception as error:
    logging.info("A terrible error happened: %s", error)
```

```
except Exception:
    logging.info("A terrible error happened", exc_info=True)
```

Common misuses

```
Logging.getLogger("project_name")
```

```
Logging.getLogger(__name__)
```

Common misuses

```
logger.debug("Hello {}".format(name))
```

```
logger.debug("Hello %s", name))
```

Common misuses

```
try:
    magic()
except Exception:
    logging.exception("Wops, something failed")
    raise
```

How to configure it

basicConfig

```python
import logging
logging.basicConfig(level='INFO')
```

filename
filemode
format
datefmt
level
stream

TechAtBloomberg.com

Bloomberg

Engineering

dictConfig

```
config = {    # logging.config.dictConfig(config)
    'disable_existing_loggers': False,
    'version': 1,
    'formatters': {
        'short': {
            'format': '%(asctime)s %(levelname)s %(name)s: %(message)s’
        },
    },
    'handlers': {
        'console': {
            'level': 'INFO',
            'formatter': 'short',
            'class': 'logging.StreamHandler',
        },
    },
    'loggers': {
        '': {
            'handlers': ['console'],
            'level': 'INFO',
        },
        'plugins’: {
            'level’: ‘ERROR',
        }
    },
}
```

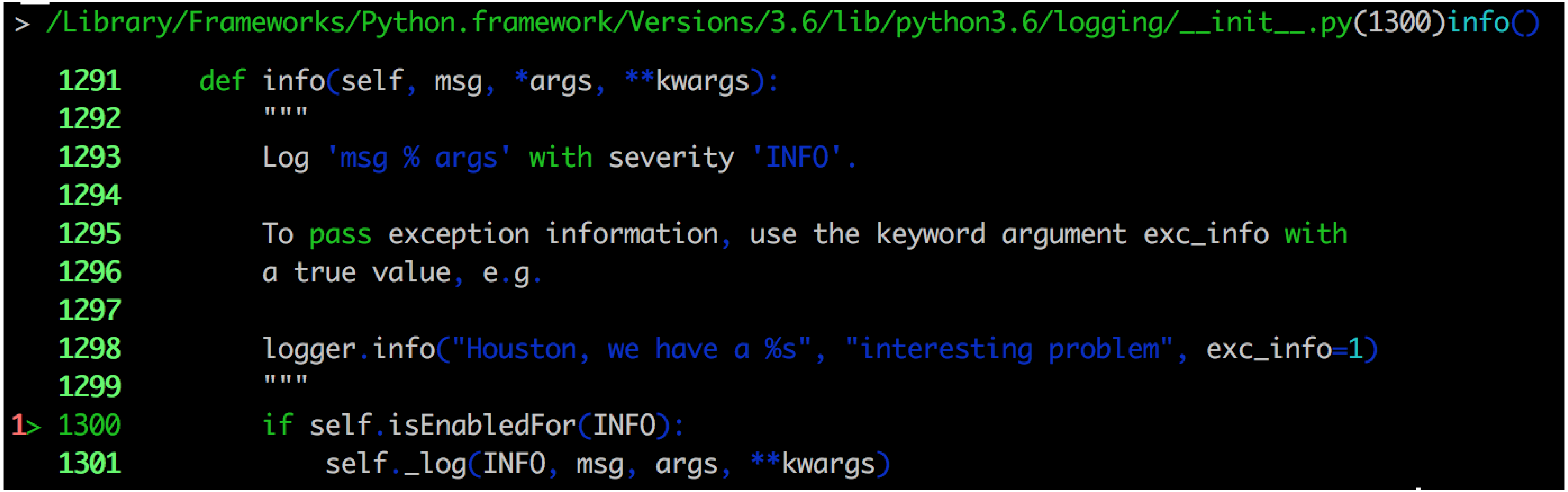
Bloomberg

Engineering

Show me the code!

```
import logging
logging.basicConfig(level='INFO')
logger = logging.getLogger(’logname')
logger.info('Hello %s', PiterPy')
```

Logger.info



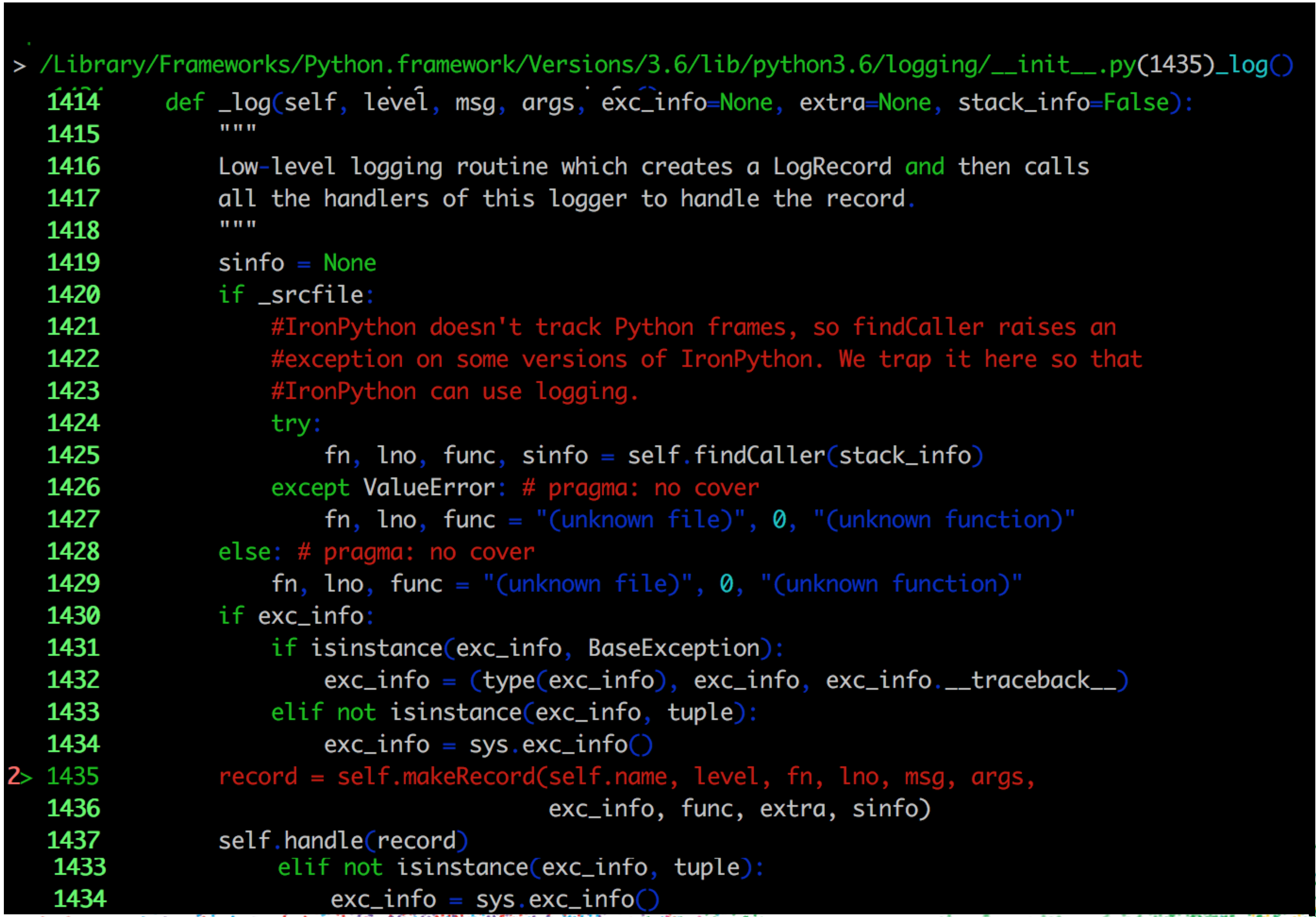
```
> /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/logging/__init__.py(1300)info()

    def info(self, msg, *args, **kwargs):
        """
        Log 'msg % args' with severity 'INFO'.

        To pass exception information, use the keyword argument exc_info with
        a true value, e.g.

        logger.info("Houston, we have a %s", "interesting problem", exc_info=1)
        """
1>      if self.isEnabledFor(INFO):
            self._log(INFO, msg, args, **kwargs)
```

TechAtBloomberg.com

Bloomberg Engineering

Show me the code

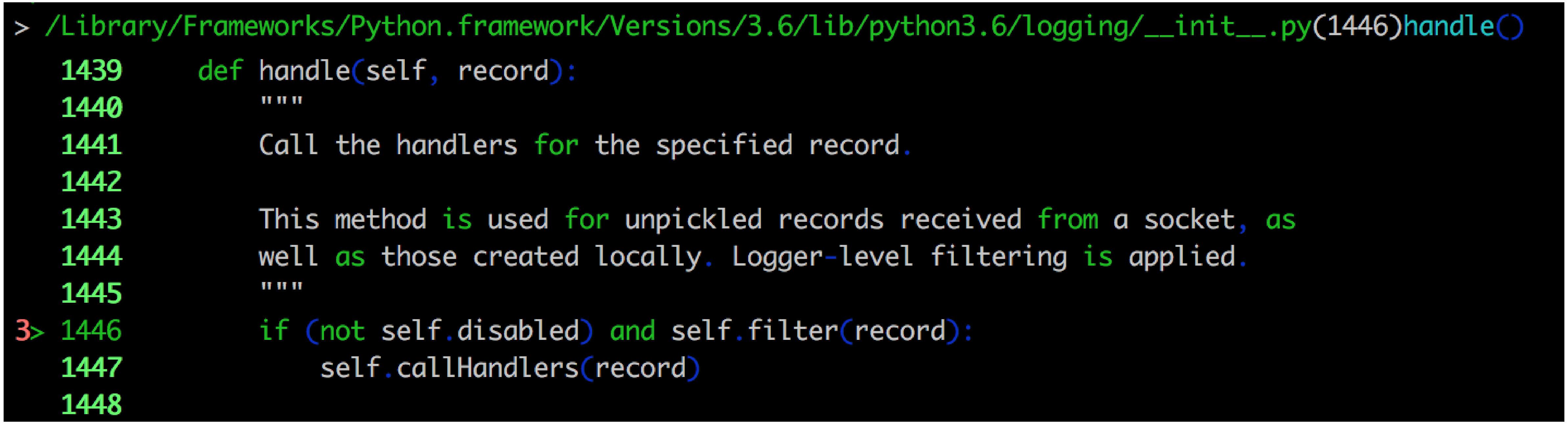
Logger._log



```
> /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/logging/__init__.py(1435)_log()
   1414      def _log(self, level, msg, args, exc_info=None, extra=None, stack_info=False):
   1415          """
   1416          Low-level logging routine which creates a LogRecord and then calls
   1417          all the handlers of this logger to handle the record.
   1418          """
   1419          sinfo = None
   1420          if _srcfile:
   1421              #IronPython doesn't track Python frames, so findCaller raises an
   1422              #exception on some versions of IronPython. We trap it here so that
   1423              #IronPython can use logging.
   1424              try:
   1425                  fn, lno, func, sinfo = self.findCaller(stack_info)
   1426              except ValueError: # pragma: no cover
   1427                  fn, lno, func = "(unknown file)", 0, "(unknown function)"
   1428          else: # pragma: no cover
   1429              fn, lno, func = "(unknown file)", 0, "(unknown function)"
   1430          if exc_info:
   1431              if isinstance(exc_info, BaseException):
   1432                  exc_info = (type(exc_info), exc_info, exc_info.__traceback__)
   1433              elif not isinstance(exc_info, tuple):
   1434                  exc_info = sys.exc_info()
2> 1435          record = self.makeRecord(self.name, level, fn, lno, msg, args,
   1436                                   exc_info, func, extra, sinfo)
   1437          self.handle(record)
   1433              elif not isinstance(exc_info, tuple):
   1434                  exc_info = sys.exc_info()
```

Show me the code

Logger.handle



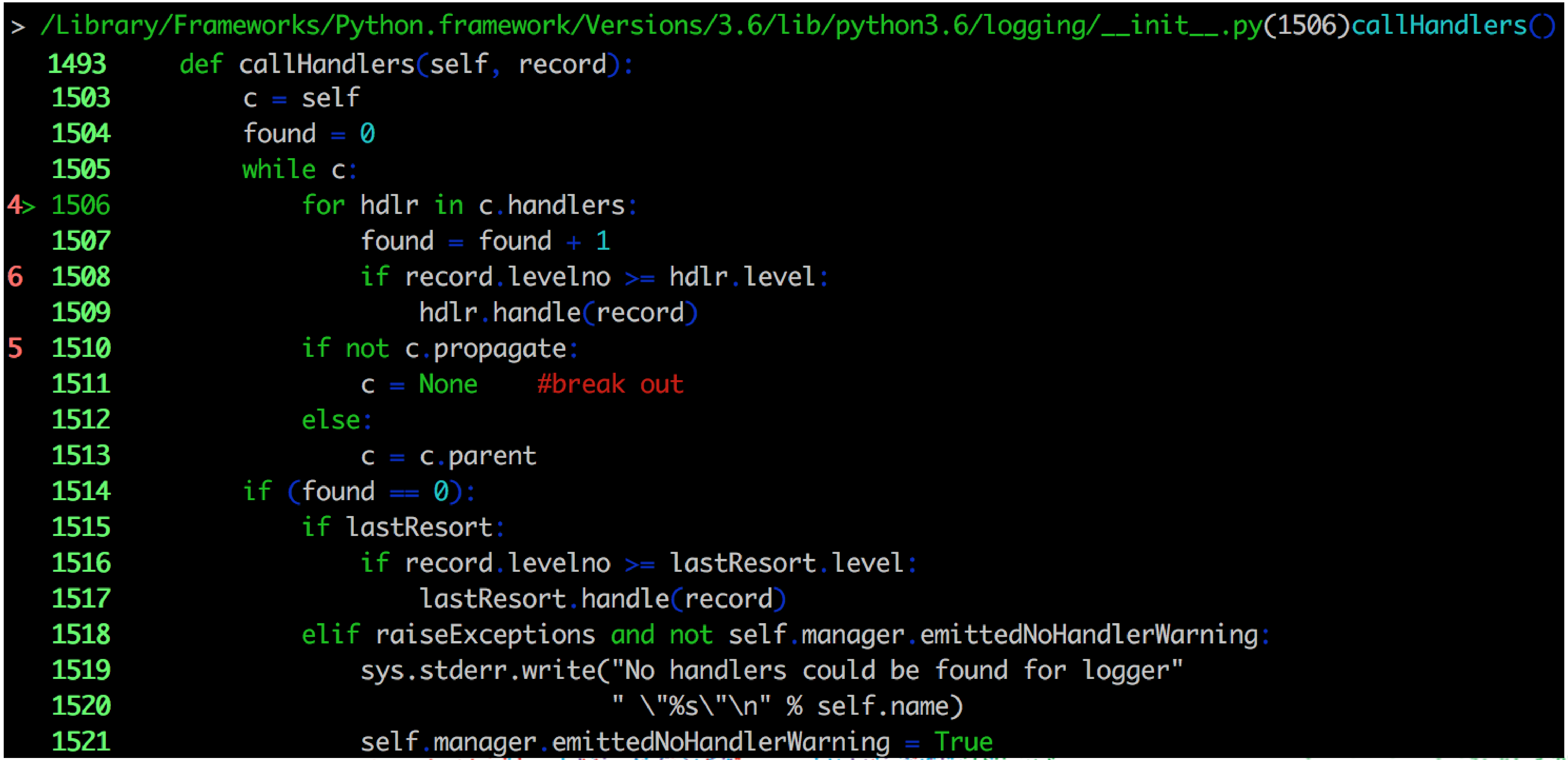
```
> /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/logging/__init__.py(1446)handle()
   1439         def handle(self, record):
   1440             """
   1441             Call the handlers for the specified record.
   1442
   1443             This method is used for unpickled records received from a socket, as
   1444             well as those created locally. Logger-level filtering is applied.
   1445             """
3> 1446             if (not self.disabled) and self.filter(record):
   1447                 self.callHandlers(record)
   1448
```

TechAtBloomberg.com

Bloomberg

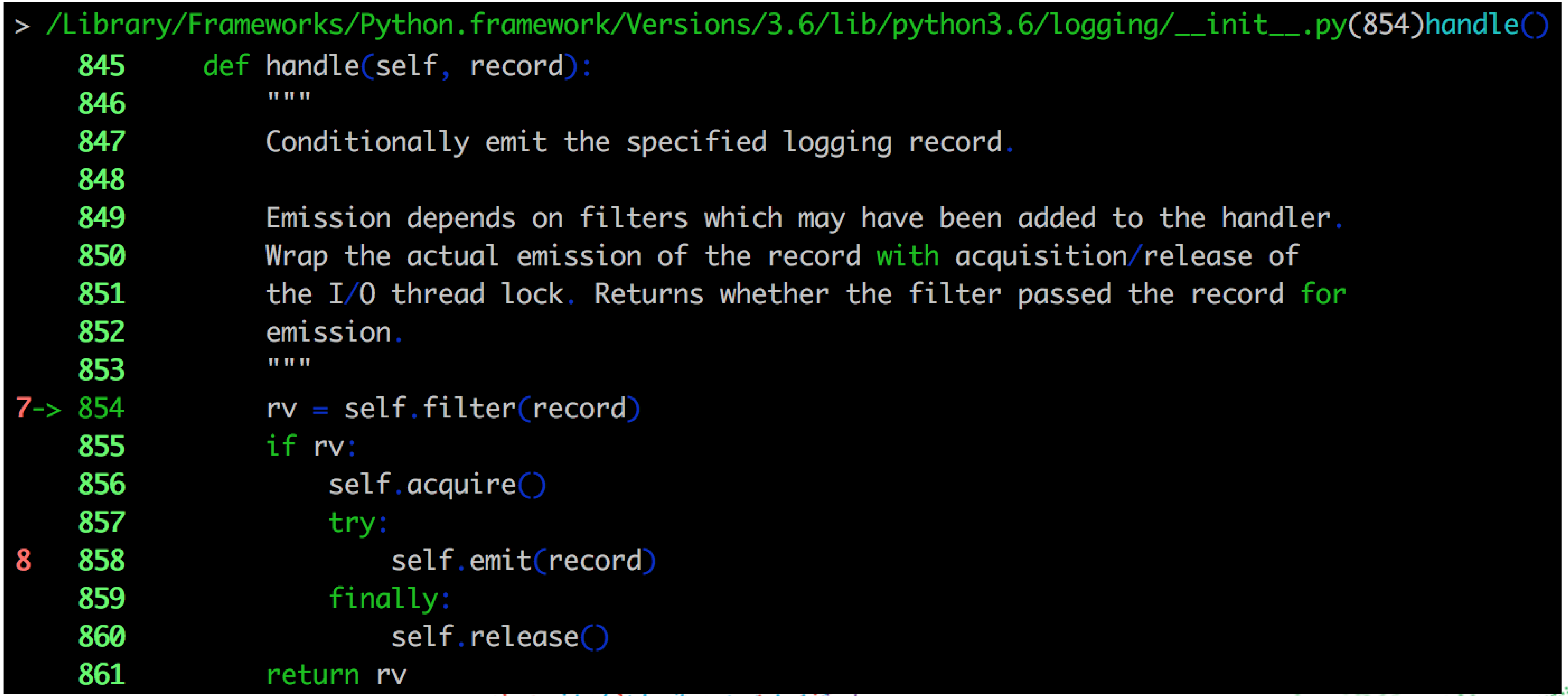
Engineering

Logger.callHandlers



```
> /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/logging/__init__.py(1506)callHandlers()
   1493        def callHandlers(self, record):
   1503            c = self
   1504            found = 0
   1505            while c:
4> 1506                for hdlr in c.handlers:
   1507                    found = found + 1
6  1508                    if record.levelno >= hdlr.level:
   1509                        hdlr.handle(record)
5  1510                if not c.propagate:
   1511                    c = None    #break out
   1512                else:
   1513                    c = c.parent
   1514            if (found == 0):
   1515                if lastResort:
   1516                    if record.levelno >= lastResort.level:
   1517                        lastResort.handle(record)
   1518                elif raiseExceptions and not self.manager.emittedNoHandlerWarning:
   1519                    sys.stderr.write("No handlers could be found for logger"
   1520                                     " \"%s\"\n" % self.name)
   1521                    self.manager.emittedNoHandlerWarning = True
```

Handler.handle



```
> /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/logging/__init__.py(854)handle()
    845         def handle(self, record):
    846             """
    847             Conditionally emit the specified logging record.
    848
    849             Emission depends on filters which may have been added to the handler.
    850             Wrap the actual emission of the record with acquisition/release of
    851             the I/O thread lock. Returns whether the filter passed the record for
    852             emission.
    853             """
7-> 854             rv = self.filter(record)
    855             if rv:
    856                 self.acquire()
    857                 try:
8   858                     self.emit(record)
    859                 finally:
    860                     self.release()
    861             return rv
```

Sample recipes

Multiple handlers

```
'loggers': {
    'request': {
        'handlers': ['request_logs'],
        'level’: ‘DEBUG',
        'propagate': False,
    },
    '': {
        'handlers': ['info_log_file', 'error_log_file', 'debug_log_file', 'mail_admins'],
        'level': 'DEBUG',
    },
}
```

Logging JSON

```
import logging
import logging.config
import json
ATTR_TO_JSON = ['created', 'filename', 'funcName', 'levelname', 'lineno', 'module', 'msecs', 'msg',
'name', 'pathname', 'process', 'processName', 'relativeCreated', 'thread', 'threadName']
class JsonFormatter:
    def format(self, record):
        obj = {attr: getattr(record, attr)
                  for attr in ATTR_TO_JSON}
        return json.dumps(obj, indent=4)

handler = logging.StreamHandler()
handler.formatter = JsonFormatter()
logger = logging.getLogger(__name__)
logger.addHandler(handler)
logger.error("Hello")
```

Changing the LogRecord factory

```
import logging
old_factory = logging.getLogRecordFactory()
counter = 0
def record_factory(*args, **kwargs):
        global counter
        counter += 1
        record = old_factory(*args, **kwargs)
        record.counter = counter
        return record

logging.setLogRecordFactory(record_factory)
logging.basicConfig(level="INFO", format="%(asctime)-15s %(counter)s %(message)s")
logging.info("First log")
logging.info("Second log")
```

Buffering

```
import logging
import logging.handlers

class SmartBufferHandler(logging.handlers.MemoryHandler):
    … init ...
    def emit(self, record):
        if len(self.buffer) == self.capacity - 1:
            self.buffer.pop(0)
            super().emit(record)
handler = SmartBufferHandler(buffered=2, target=logging.StreamHandler(), flushLevel=ERROR)
logger = logging.getLogger(__name__)
logger.setLevel("INFO")
logger.addHandler(handler)

logging.getLogger(__name__).error("Hello1")
logging.getLogger(__name__).info("Hello2")
logging.getLogger(__name__).info("Hello3")
logging.getLogger(__name__).error("Hello4")
```

Non-blocking handling of records

```
que = queue.Queue(-1) # no limit on size

queue_handler = QueueHandler(que)
handler = logging.StreamHandler()
listener = QueueListener(que, handler)

root = logging.getLogger()
root.addHandler(queue_handler)

listener.start()
root.warning('Look out!')
listener.stop()
```

Console output

```
class MaxLevelFilter:
    def __init__(self, max_level=None):
        self.max_level = max_level
    def filter(self, record):
        return record.levelno <= self.max_level

import sys
logger = logging.getLogger()
logger.setLevel(logging.DEBUG)
stdout = logging.StreamHandler(sys.stdout)
stdout.setLevel("DEBUG")
stdout.addFilter(MaxLevelFilter("INFO"))
stderr = logging.StreamHandler(sys.stderr)
stderr.setLevel(logging.WARNING)
logger.addHandler(stdout)
logger.addHandler(stderr)

logger.info('INFO')  # to stdout only
logger.error('ERROR')  # to stderr
```

TechAtBloomberg.com

Bloomberg Engineering

Combining it with argparse

```
import logging
import sys
import argparse
parser = argparse.ArgumentParser()
levels = ["ERROR", "WARNING", "INFO", "DEBUG"]
parser.add_argument("-v", "--verbosity", action="count", default=0)
parser.add_argument('--outfile', nargs='?', type=argparse.FileType('w'),
                    default=sys.stdout)
args = parser.parse_args()
level = levels[args.verbosity]
logging.basicConfig(level=level, stream=args.outfile)
```

Lazy evaluation

```
import logging
def expensive_call():
    print("This was really expensive")
    return "Hi”

class LazyLog:
    def __init__(self, func, *args, **kwargs):
        self.func = func
        self.args = args
        self.kwargs = kwargs
    def __str__(self):
        return self.func(*self.args, **self.kwargs) or None

logging.basicConfig(level="INFO")
logging.debug(LazyLog(expensive_call))
logging.info(LazyLog(expensive_call))
```

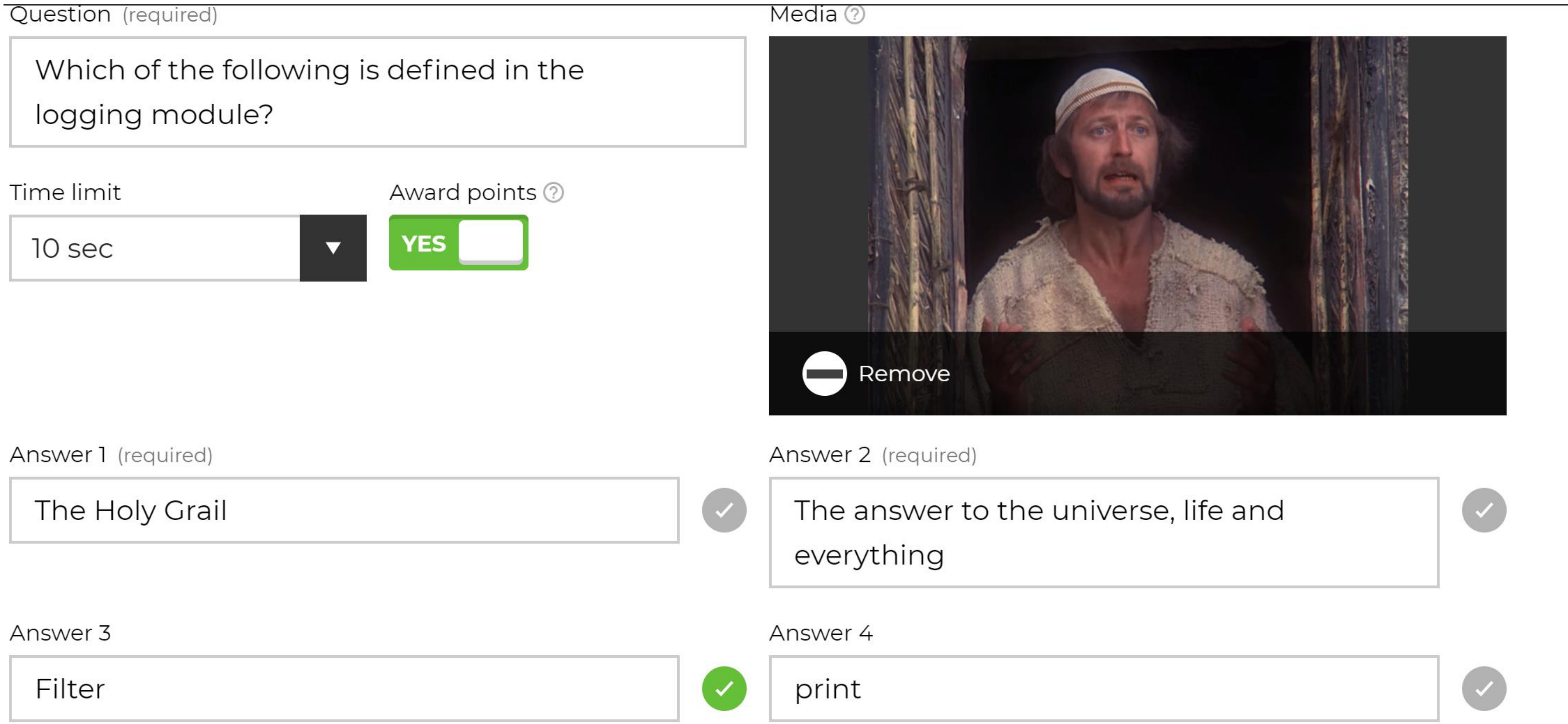
Quiz Time

- Go to kahoot.it
- Code: <questions in next slides>

TechAtBloomberg.com

Bloomberg

Engineering



Question (required)

Which of the following is defined in the logging module?

Media

Time limit: 10 sec

Award points: YES

Answer 1 (required): The Holy Grail

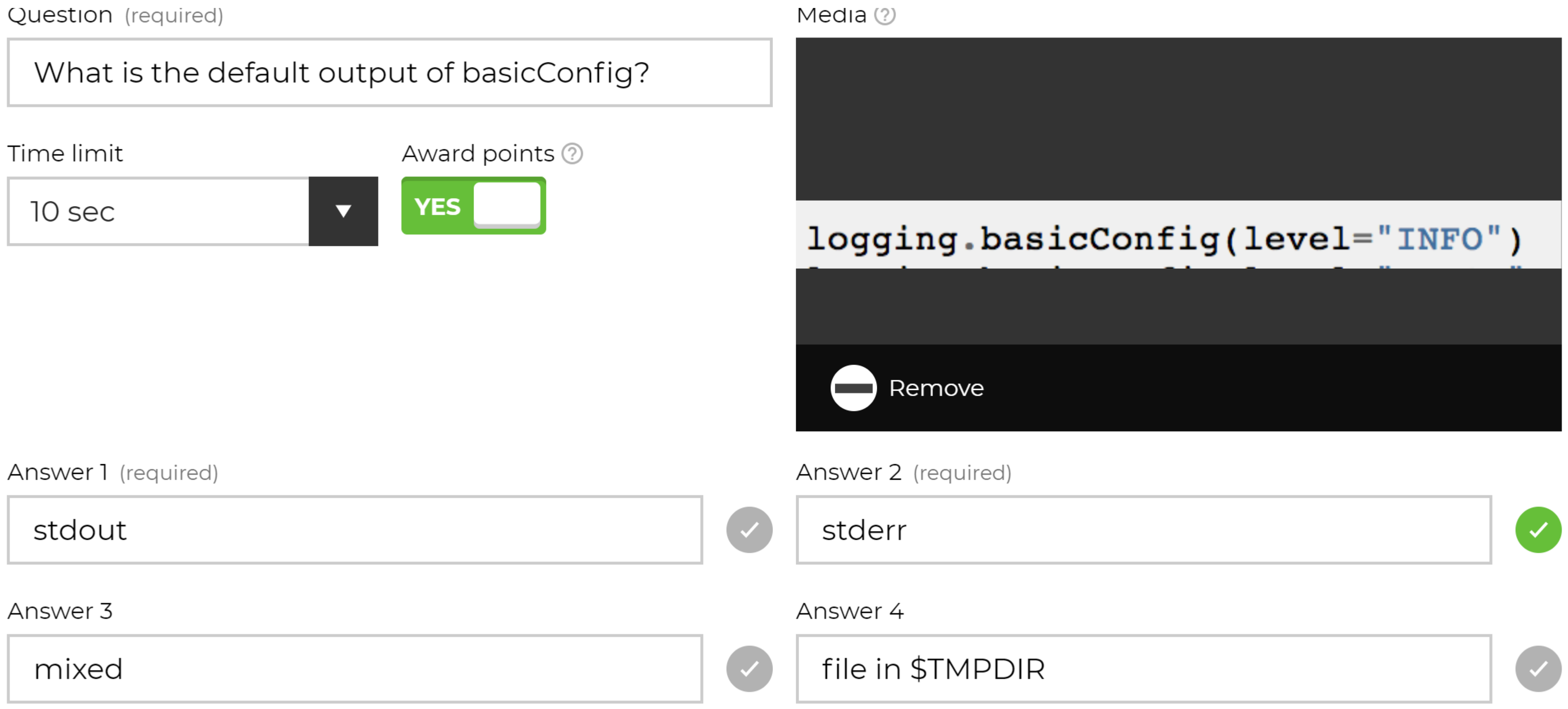
Answer 2 (required): The answer to the universe, life and everything

Answer 3: Filter ✓

Answer 4: print

TechAtBloomberg.com

Bloomberg Engineering



Question (required)

What is the default output of basicConfig?

Media

```
logging.basicConfig(level="INFO")
```

Time limit: 10 sec

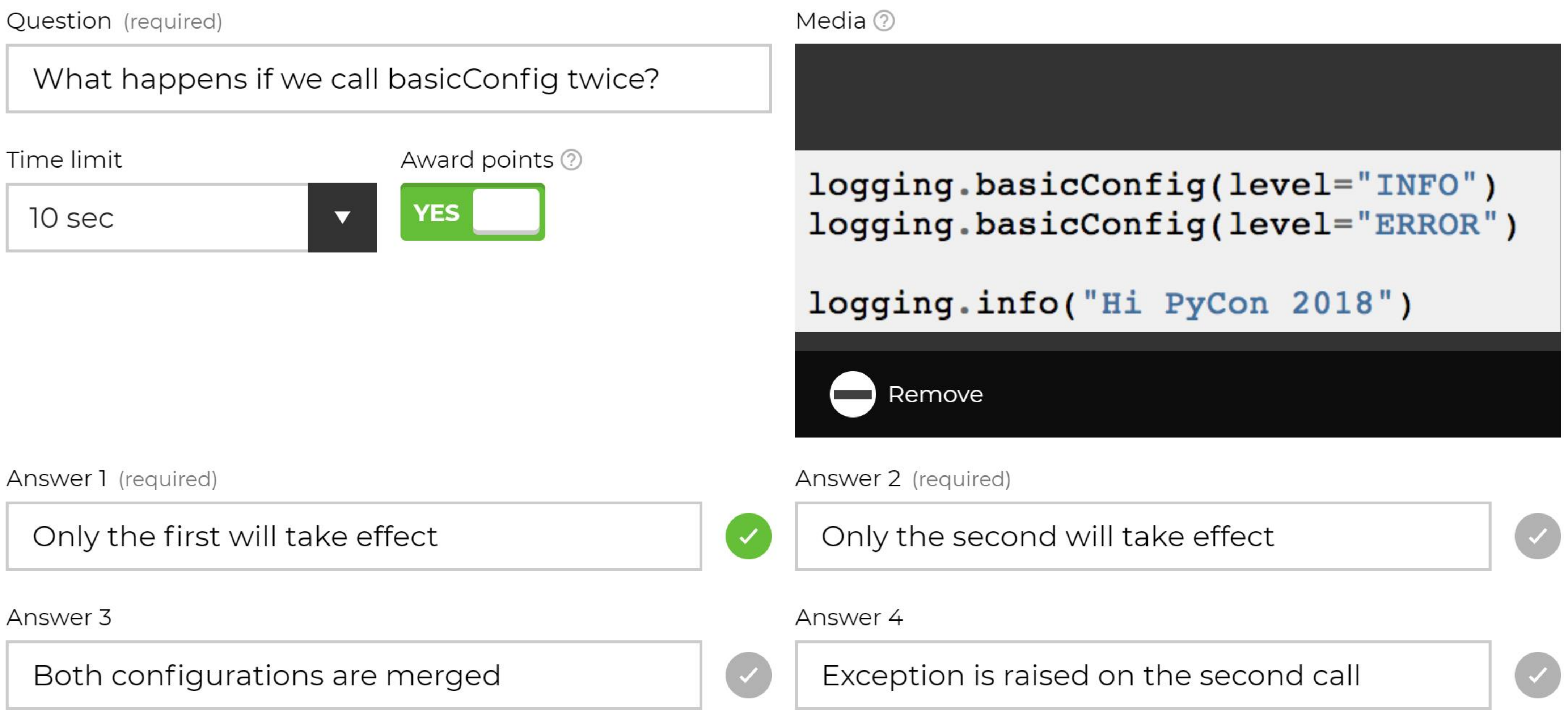
Award points: YES

Answer 1 (required)	Answer 2 (required)
stdout	[✓ stderr]
Answer 3	Answer 4
mixed	file in $TMPDIR

TechAtBloomberg.com

Bloomberg

Engineering



Question (required)

What happens if we call basicConfig twice?

Time limit

10 sec ▼

Award points ?

YES

Media ?

```
logging.basicConfig(level="INFO")
logging.basicConfig(level="ERROR")

logging.info("Hi PyCon 2018")
```

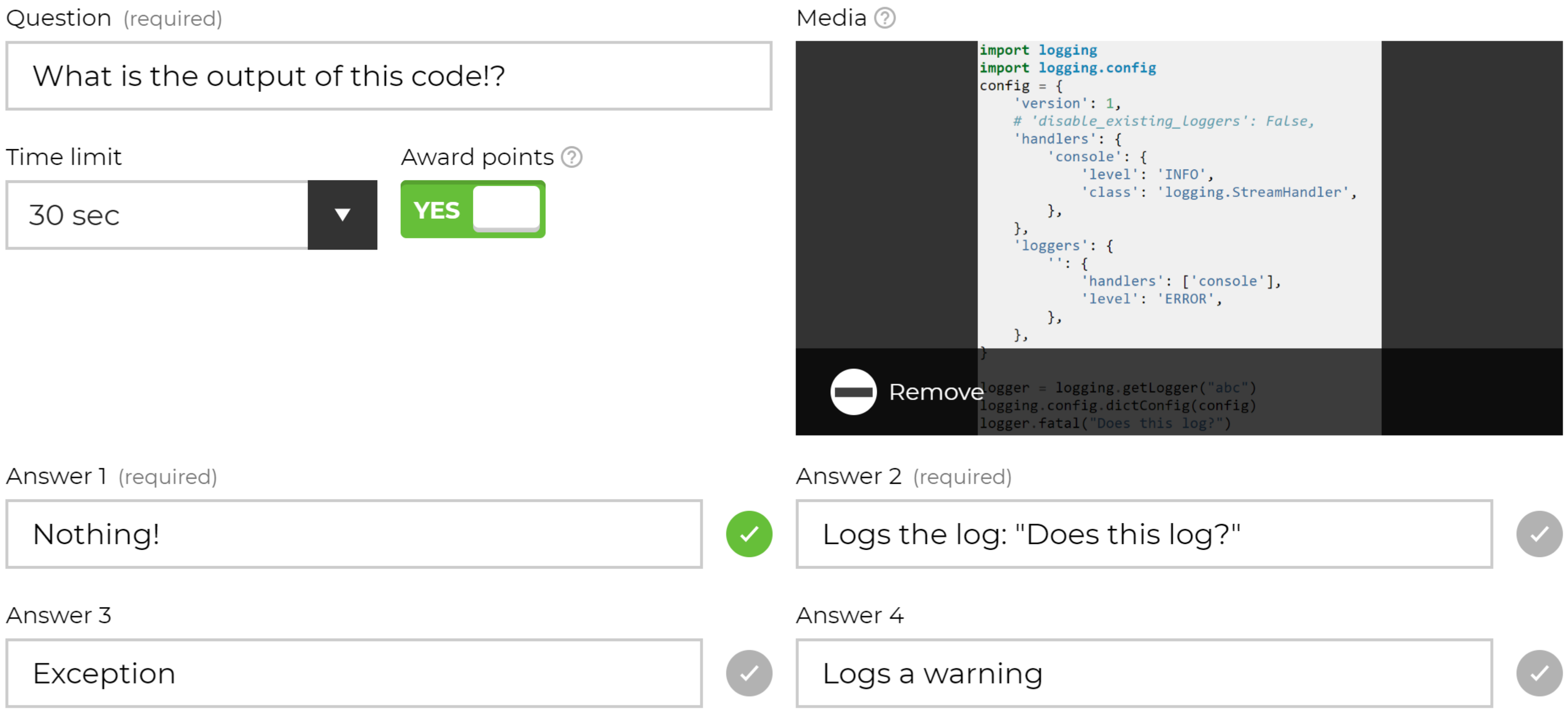
Remove

Answer 1 (required)	Answer 2 (required)
Only the first will take effect ✓ (chosen)	Only the second will take effect ✓
Answer 3	Answer 4
Both configurations are merged ✓	Exception is raised on the second call ✓

TechAtBloomberg.com

Bloomberg

Engineering



Question (required)

What is the output of this code!?

Time limit

30 sec ▼

Award points

YES

Media

```
import logging
import logging.config
config = {
    'version': 1,
    # 'disable_existing_loggers': False,
    'handlers': {
        'console': {
            'level': 'INFO',
            'class': 'logging.StreamHandler',
        },
    },
    'loggers': {
        '': {
            'handlers': ['console'],
            'level': 'ERROR',
        },
    },
}

logger = logging.getLogger("abc")
logging.config.dictConfig(config)
logger.fatal("Does this log?")
```

Remove

Answer 1 (required)

Nothing! ✓

Answer 2 (required)

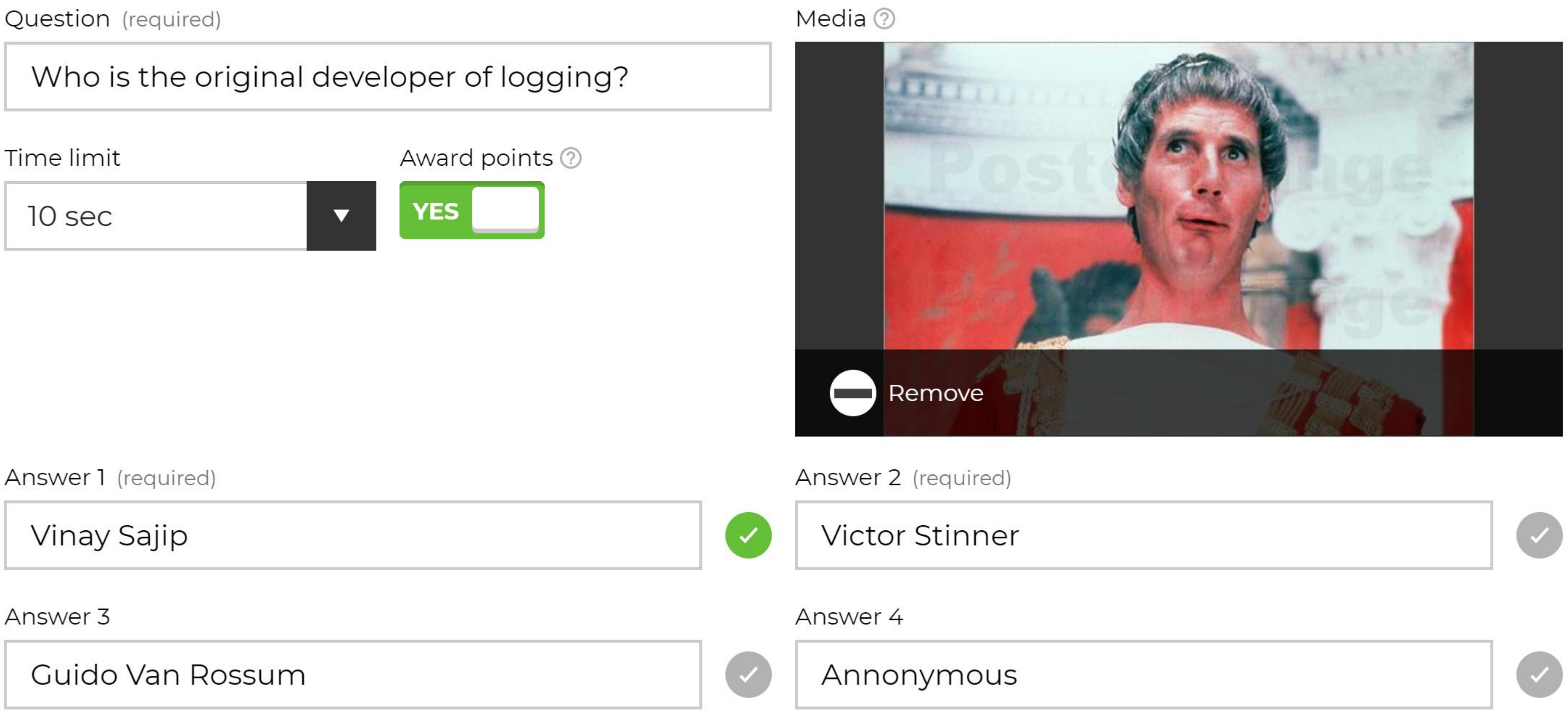
Logs the log: "Does this log?"

Answer 3

Exception

Answer 4

Logs a warning



Question (required)

Who is the original developer of logging?

Time limit: 10 sec

Award points: YES

Media

Answer 1 (required)	Answer 2 (required)
Vinay Sajip ✓	Victor Stinner
Answer 3	Answer 4
Guido Van Rossum	Annonymous

TechAtBloomberg.com

Bloomberg

Engineering

Links

Some useful links about the talk:

- https://opensource.com/article/17/9/python-logging
- https://docs.python.org/3.7/howto/logging.html
- https://docs.python.org/3/howto/logging-cookbook.html
- https://github.com/python/cpython/blob/master/Lib/logging/__init__.py

Благодарю вас!

Questions?

TechAtBloomberg.com