



ĐẠI HỌC ĐÀ NẴNG

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

# LOGGING REACTIVE PROGRAMMING NETWORK PROGRAMMING



**Khoa Công nghệ thông tin**

ThS. Nguyễn Thị Lệ Quyên

# Nội dung môn học

- Cơ bản ngôn ngữ lập trình Python
- **Logging, Reactive Programming, Network Programming**
- Tương tác Cơ sở dữ liệu
- Django framework
- Kiến thức bổ sung



ĐẠI HỌC ĐÀ NẴNG

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

Khoa CÔNG NGHỆ THÔNG TIN

ThS. Nguyễn Thị Lệ Quyên



**LOGGING**

D  
BACH KHOA

N  
A  
N  
G

# Why Log?

- Chuẩn đoán lỗi
- Xác định hành vi bất thường hoặc bất ngờ
- Xác định các vấn đề về hiệu suất hoặc năng lực
- Đối phó với các hành vi tấn công
- Tuân thủ quy định hoặc pháp luật

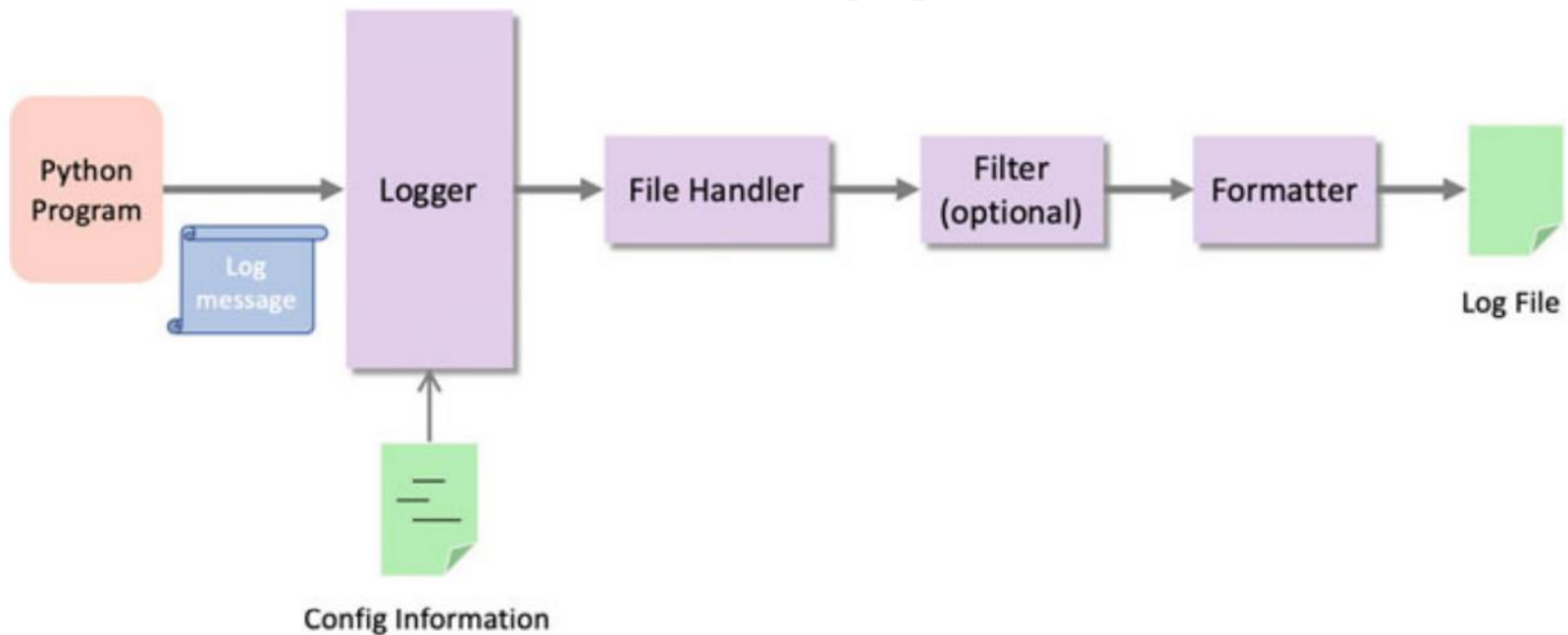
# What should log and what not?

- Log đủ thông tin
  - Vấn đề gì đã xảy ra? Xảy ra khi nào? Xảy ra ở đâu?
- Không log bất cứ thông tin cá nhân, thông tin nhạy cảm liên quan đến định danh cá nhân
  - Ví dụ: tài khoản, mật khẩu, email, sinh nhật, passport, CCCD, địa chỉ, số điện thoại, tài khoản mạng xã hội, ...
- Cẩn thận khi log trực tiếp dữ liệu đầu vào (data input) vào 1 log file.
  - Ref: [Log Injection Software Attack | OWASP Foundation](#)
- Lựa chọn thông tin khi log trong môi trường production
- Lựa chọn đúng nơi in log
  - Nhiều ứng dụng chia ra in log chung vào 1 file, errors và exceptions vào 1 file khác, thông tin liên quan tới bảo mật vào 1 file khác

# Sao không chỉ sử dụng lệnh print?

- Hàm print() chỉ xuất thông tin tới stdout và stderr -> console/terminal
- Hàm print() sẽ in ra tất cả các thông tin
  - Tuy nhiên thông tin có **log level** trong từng trường hợp/ môi trường
- Thay vì chỉ cần thay đổi configuration file, nếu chỉ sử dụng hàm print() sẽ phải sửa mã code để thay đổi *log level*
- Hàm print() cho phép lập trình viên có thể sử dụng bất kỳ định dạng mà họ thích => thông tin log không được chuẩn hóa, không chung quy tắc

# Logging Module



# Logger

```
import logging

logger = logging.getLogger()

logger.error('This should be used with something
            unexpected')
```



# Kiểm soát lượng thông tin đã log


- Có 6 mức độ khác nhau (log level)
  - NOTSET – Logging bị tắt
  - DEBUG – Cung cấp thông tin chi tiết, sử dụng khi lập trình viên đang tìm kiếm và chẩn đoán errors/issues
  - INFO – Cung cấp ít thông tin hơn so với DEBUG, cung cấp thông tin đảm bảo ứng dụng đang hoạt động như mong đợi
  - WARNING – Cung cấp thông tin về 1 sự kiện không mong muốn hoặc dấu hiệu vấn đề có thể xảy ra mà lập trình viên hoặc quản trị hệ thống có thể muốn điều tra thêm
  - ERROR – Cung cấp thông tin về sự cố, vấn đề quan trọng khiến hệ thống không thể hoạt động chính xác
  - CRITICAL – Cấp độ cao nhất, dành riêng cho các tình huống quan trọng, như là chương trình không thể tiếp tục hoạt động

# Logging

```
import logging

logger = logging.getLogger()

logger.debug('This is to help with debugging')
logger.info('This is just for information')
logger.warning('This is a warning!')
logger.error('This should be used with something
              unexpected')
logger.critical('Something serious')
```

 Increasing log levels	CRITICAL	50
	ERROR	40
	WARNING	30
	INFO	20
	DEBUG	10
	NOTSET	0

# Phương thức của Logger

- `setLevel(level)` Sets this loggers log level.
- `getEffectiveLevel()` Returns this loggers log level.
- `isEnabledFor(level)` Checks to see if this logger is enabled for the log level specified.
- `debug(message)` logs messages at the debug level.
- `info(message)` logs messages at the info level.
- `warning(message)` logs messages at the warning level.
- `error(message)` logs messages at the error level.
- `critical(message)` logs messages at the critical level.

# Phương thức của Logger

- `exception(message)` This method logs a message at the error level.
  - However, it can only be used within an exception handler and includes a stack trace of any associated exception, for example:

```
import logging
logger = logging.getLogger()
try:
    print('starting')
    x = 1 / 0
    print(x)
except:
    logger.exception('an exception message')
print('Done')
```

- `log(level, message)` logs messages at the log level specified as the first parameter.

# Phương thức của Logger

- `addFilter(filter)` This method adds the specified filter filter to this logger.
- `removeFilter(filter)` The specified filter is removed from this logger object.
- `addHandler(handler)` The specified handler is added to this logger.
- `removeHandler(handler)` Removes the specified handler from this logger.

# Logging

```
import logging

#Set the root logger level
logging.basicConfig(level=logging.DEBUG)

# Use root (default) logger
logging.debug('This is to help with debugging')
logging.info('This is just for information')
logging.warning('This is a warning!')
logging.error('This should be used with something unexpected')
logging
```

DEBUG:root:This is to help with debugging  
INFO:root:This is just for information  
WARNING:root:This is a warning!  
ERROR:root:This should be used with something unexpected  
CRITICAL:root:Something serious

# Module Level Loggers

- Các module thường sẽ không sử dụng root logger để log thông tin, thay vào đó sẽ sử dụng logger được đặt tên (named logger) hoặc logger phân cấp độ module (module level logger).

```
logger1 = logging.getLogger('my logger')
```

- (Phổ biến) Tên module được sử dụng cho tên logger vì chỉ duy nhất 1 module tồn tại trong 1 hệ thống cụ thể

```
logger2 = logging.getLogger(__name__)
```

# Module Level Loggers

```
logger = logging.getLogger()
print('Root logger:', logger)

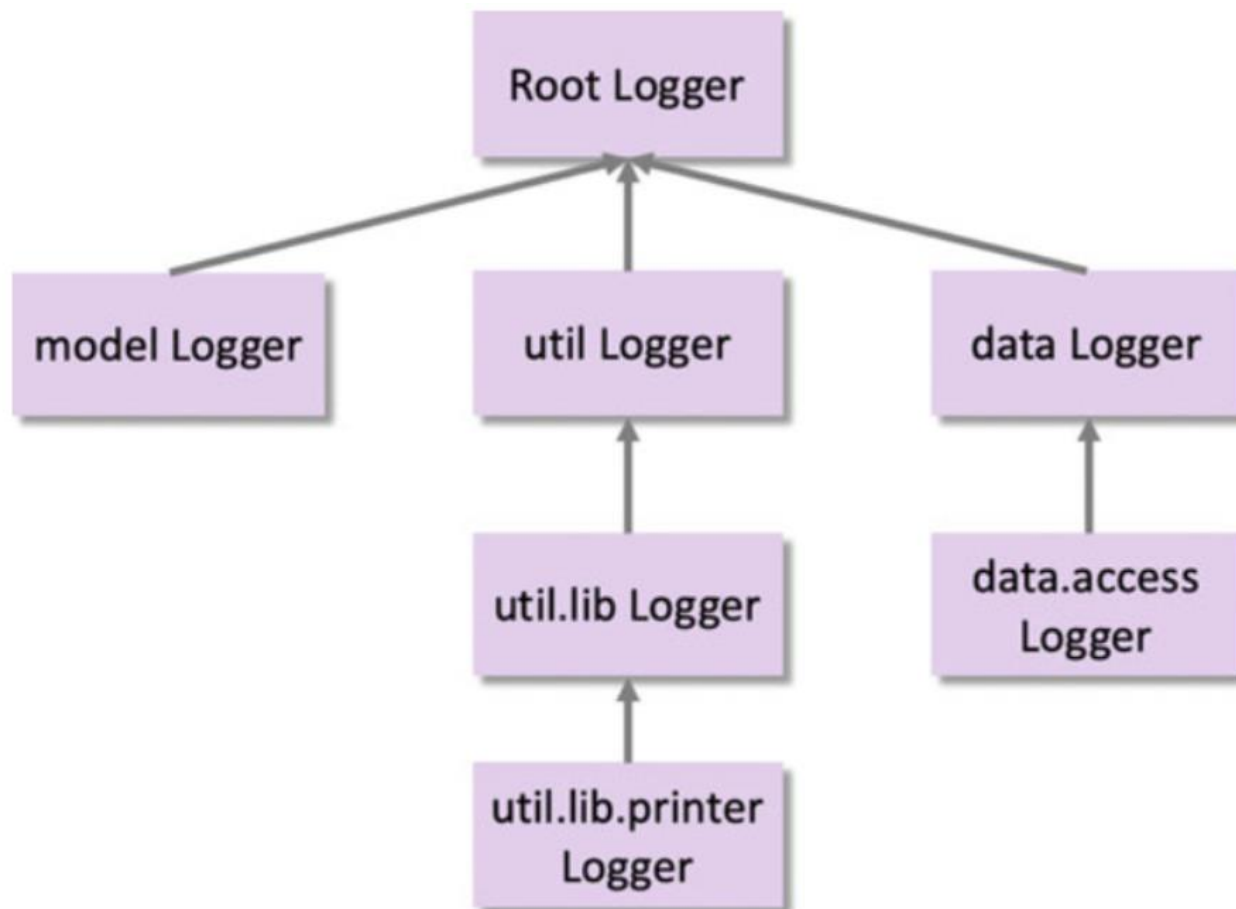
logger1 = logging.getLogger('my logger')
print('Named logger:', logger1)

logger2 = logging.getLogger(__name__)
print('Module logger:', logger2)
```

```
Root logger: <RootLogger root (WARNING)>
Named logger: <Logger my logger (WARNING)>
Module logger: <Logger __main__ (WARNING)>
```



# Logger Hierarchy



# Formatters

- Formatting Log Messages

```
logger.warning('%s is set to %d', 'count', 42)
```

- Formatting Log Output

# Formatters

- Formatting Log Output

```
import logging
logging.basicConfig(format='%(asctime)s %(message)s',
                    level=logging.DEBUG)
logger = logging.getLogger(__name__)

def do_something():
    logger.debug('This is to help with debugging')
    logger.info('This is just for information')
    logger.warning('This is a warning!')

do_something()
2022-04-07 21:19:11,720 This is to help with debugging
2022-04-07 21:19:11,720 This is just for information
2022-04-07 21:19:11,721 This is a warning!
2022-04-07 21:19:11,721 This should be used with something
unexpected
2022-04-07 21:19:11,721 Something serious
```



D  
BACH KHOA

N  
A  
N  
G