

Logging em Aplicações

Monitorando e Depurando com Eficiência

Prof. Msc. Francisco de Assis de Souza Rodrigues

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🎯 Objetivos

- Entender o papel do *logging* no ciclo de vida de uma aplicação.
- Escolher níveis de log adequados (info, warning, error, etc.).
- Implementar logs de forma eficiente e padronizada.
- Utilizar ferramentas para análise e monitoramento de logs.
- Evitar armadilhas comuns no uso de logs.

O que é Logging?

Logging é o processo de registrar eventos que ocorrem durante a execução de um software.

Ele é fundamental para:

- Monitorar o comportamento da aplicação em produção.
- Depurar erros de forma mais eficiente.
- Analisar métricas e uso da aplicação.
- Cumprir requisitos legais ou de auditoria.

Níveis de Log

Nível	Descrição	Exemplo
DEBUG	Detalhes de execução, útil para desenvolvimento	DEBUG: Resultado do cálculo: 42
INFO	Informações sobre o fluxo normal da aplicação	INFO: Usuário logado com sucesso
WARNING	Algo inesperado, mas a aplicação continua	WARNING: Tentativa de login com senha incorreta
ERROR	Erro que impede a ação atual	ERROR: Falha ao conectar ao banco de dados
CRITICAL	Erro grave, pode derrubar a aplicação	CRITICAL: Servidor fora do ar

Boas Práticas

- Nunca logar senhas, tokens ou dados sensíveis.
- Use identificadores em logs (ex: ID do usuário).
- Use uma biblioteca de logging (não `print()`).
- Adote um formato estruturado (JSON é bom para integração com sistemas de log).
- Evite excesso de log em produção (custo e desempenho).
- Separe logs por ambiente: dev, staging, production.

Logging

```python
import logging

# Configuração básica
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s [%(levelname)s] %(message)s',
    filename='app.log',  # ou use StreamHandler para console
    filemode='a'
)

logging.debug("Isso é um log de debug.")
logging.info("Aplicação iniciada com sucesso.")
logging.warning("Recurso depreciado usado.")
logging.error("Erro ao acessar banco de dados.")
logging.critical("Falha crítica: servidor indisponível.")
```

Logging

```
const winston = require('winston');

const logger = winston.createLogger({
  level: 'info',
  format: winston.format.combine(
    winston.format.timestamp(),
    winston.format.json()
  ),
  transports: [new winston.transports.Console()]
});

logger.info("Aplicação iniciada");
logger.warn("Senha fraca detectada");
logger.error("Erro ao salvar no banco");
```

🔍 Ferramentas de Logging

📌 Sentry

- Monitoramento de exceções em tempo real.
- Útil para front e backend.
- Captura *stack trace*, ambiente e usuário.

https://sentry.io/

🔍 Ferramentas de Logging

📌 ELK Stack (Elasticsearch + Logstash + Kibana)

- Agregador de logs com dashboards e filtros avançados.
- Muito usado em ambientes corporativos.

https://www.elastic.co/elastic-stack

🔍 Ferramentas de Logging

📌 Loki + Grafana

- Alternativa moderna e leve ao ELK.
- Ideal para microserviços e Kubernetes.

https://grafana.com/oss/loki/

🔍 Ferramentas de Logging

📌 Loggly / Papertrail / Datadog

- Soluções SaaS para logs com análise e alertas.

https://www.datadoghq.com/

https://www.papertrail.com/

🧪 Experimento Prático (Hands-on)

📋 Instruções

1. Execute cada célula no Google Colab.

2. Analise os logs gerados no console.

3. Adapte os exemplos para o seu próprio projeto se quiser.

https://colab.research.google.com/drive/18QDEH3JJYTRPjxXrp1zsXNHkF8Dzv2i2?usp=sharing



📌 Tarefa para a próxima aula

Desafio: Refatore sua aplicação do PI que você já tenha feito (ou a do hands-on) adicionando um sistema de logging completo.

Use:

- Logs com INFO, WARNING, ERROR
- Identificação de usuário/session
- Logs salvos em arquivo ou integrados com alguma ferramenta (ex: Sentry ou Loki)

Entrega: Enviar link do repositório no GitHub + prints do log funcionando.

🧠 Reflexão

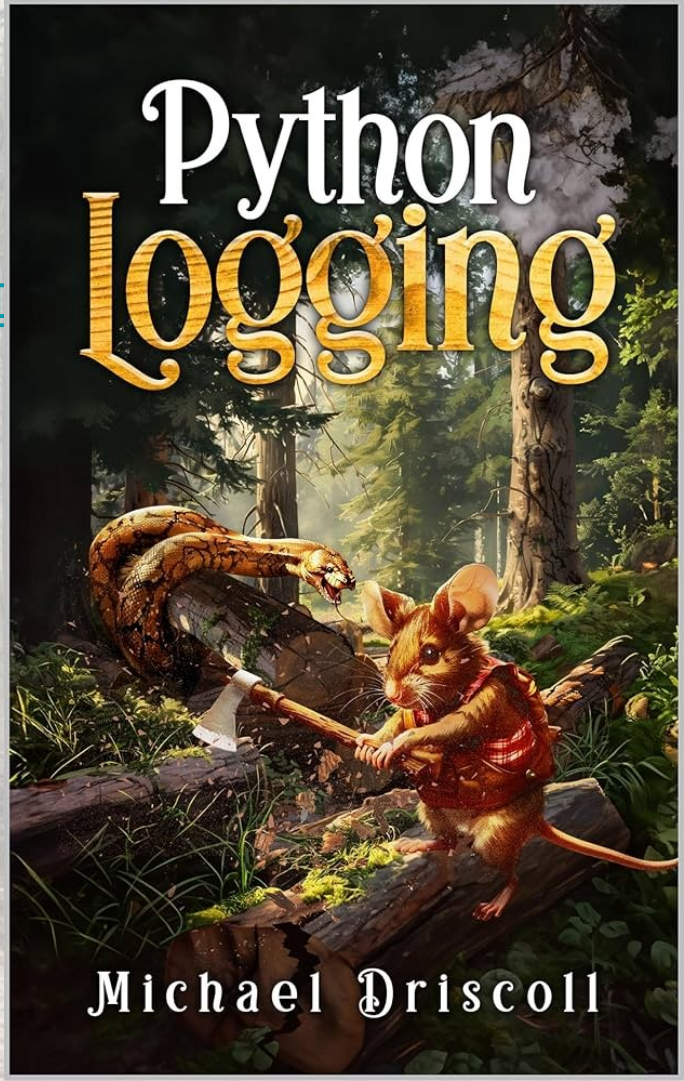
“Aquilo que não se pode medir, não se pode melhorar.” – Lord Kelvin

Logging é como a caixa-preta de uma aplicação: quanto mais bem configurada, melhor conseguimos entender, melhorar e antecipar falhas.



Bibliografia

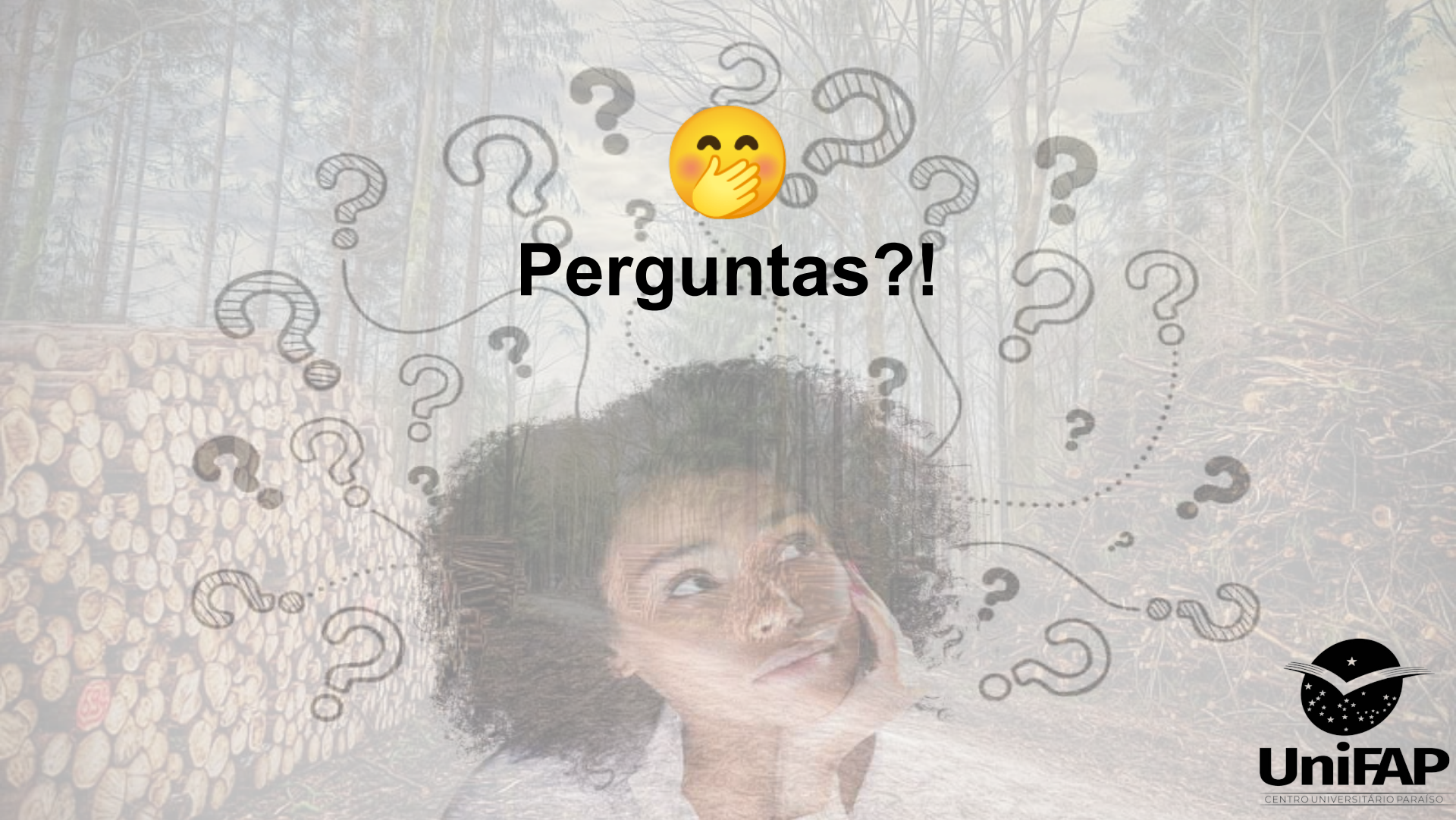
https://www.amazon.com.br/Python-Logging-Auditing-D
dp/B0D4MKTLYW



Referências

CHATGPT. **Fundamentos de Inteligência Artificial**. OpenAI, 2025. Disponível em: https://chat.openai.com. Acesso em: *20/03/2025*.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO


Perguntas?!

Eclesiastes 3:5 – "Tempo de espalhar pedras e tempo de ajuntar pedras."

Em alguns momentos será preciso plantar; em outros, colher. Respeite seus ciclos.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO