INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA - GOIÁS
CAMPUS GOIÂNIA
PÓS-GRADUAÇÃO EM INTELIGÊNCIA ARTIFICIAL APLICADA
DISCIPLINA: LINGUAGEM DE PROGRAMAÇÃO APLICADA
PROFESSOR: OTÁVIO CALAÇA XAVIER
DATA: 29/05/2025

Seminários sobre Python: Técnicas e Abordagens Atuais

Formação dos Grupos

- Cada grupo pode conter de **3 a 4 estudantes**.
- Enviar, via fórum no Moodle: **composição do grupo** e **tema escolhido**. **Não selecionar tema já escolhido por outro grupo.**
- Todos os grupos apresentarão **no mesmo dia** (a definir).

Seleção de Ferramenta ou Abordagem

- Para o **tema atribuído**, o grupo escolhe **uma** ferramenta, biblioteca ou abordagem em Python (sugerida ou aprovada pelo professor) e a analisa em profundidade.

Estrutura da Apresentação

(~10 slides · 10 a 15 min por grupo)

1. **Motivação / Problema** – por que o tema importa? (≤ 2 min)
2. **Visão Geral** – propósito & características
3. **Funcionamento Interno** – arquitetura, algoritmo ou teoria
4. **Demonstração em Python** – código executável (pode ser notebook ou Colab)
5. **Vantagens, Limitações & Boas Práticas**
6. **Caso de Uso / Estudo de Caso**
7. **Conclusão**
8. **Referências** – docs, artigos, posts (formato ABNT/APA)

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA - GOIÁS
CAMPUS GOIÂNIA
PÓS-GRADUAÇÃO EM INTELIGÊNCIA ARTIFICIAL APLICADA
DISCIPLINA: LINGUAGEM DE PROGRAMAÇÃO APLICADA
PROFESSOR: OTÁVIO CALAÇA XAVIER
DATA: 29/05/2025

Entregáveis

- **Slides** (PDF)
  Prazo: até 24 h antes do seminário

- **Notebook/demo** (link Colab ou GitHub)
  Prazo: até 24 h antes

- **Relatório técnico** (6–8 páginas, PDF)
  Prazo: até 48 h após a apresentação

Avaliação (10,0 pt)

- **Profundidade técnica & correção** — 3,0 pt
- **Clareza didática & storytelling** — 2,5 pt
- **Demonstração prática funcional** — 3,0 pt
- **Qualidade dos slides / material** — 1,0 pt
- **Pontualidade & respeito ao tempo** (10–15 min) — 1,0 pt

Temas Disponíveis

1. **Ética, Privacidade & Governança de Dados**
   - Explora os princípios de uso responsável de dados em projetos Python, incluindo modelos de anonimização vs. pseudonimização, requisitos legais (LGPD/GDPR) e frameworks de auditoria. Aborda também estratégias para mitigar viés em algoritmos e manter registros seguros de acesso e alterações em pipelines de dados.
   - Ferramentas sugeridas: `opendp`, `diffprivlib`, `raiutils`, `great_expectations`

2. **Reprodutibilidade Científica: Notebook → Artigo**

    - Detalha um fluxo completo para garantir que experimentos em notebooks possam ser convertidos em artefatos publicados e replicáveis. Cobre gerenciamento de dependências, controle de versões de código e dados, configuração de seeds aleatórios e uso de ferramentas para gerar relatórios e artigos a partir de código-fonte.
    - Ferramentas sugeridas: `conda`, `poetry`, `pip-tools`, `nbconvert`, `papermill`

3. **Profiling & Otimização de Scripts de Dados**

    - Apresenta técnicas para identificar gargalos de CPU e I/O em scripts Python, usando ferramentas de profiling (e.g. cProfile, line_profiler) e benchmarking (timeit). Discute como aplicar vetorização, paralelismo leve e ajustes de memória para acelerar processamentos de grandes volumes de dados.
    - Ferramentas sugeridas: `timeit`, `cProfile`, `line_profiler`, `memory_profiler`, `joblib`

4. **Testes Automatizados em Pipelines de Dados**

    - Mostra como estruturar uma suíte de testes para ETL e fluxos de análise, incluindo testes unitários, de integração e de esquema. Enfatiza a criação de dados sintéticos (fakes/mocks), uso de frameworks de teste (e.g. pytest, hypothesis) e integração contínua para garantir a qualidade durante o desenvolvimento.
    - Ferramentas sugeridas: `pytest`, `hypothesis`, `faker`, `unittest.mock`, *CI* (GitHub Actions)

5. **Storytelling de Dados: Narrativas Visuais Eficazes**

    - Aborda a arte de transformar resultados de análise em histórias visuais claras e persuasivas. Cobre estruturas narrativas (pirâmide invertida, jornada do usuário), escolha de tipos de gráfico e elementos visuais (cores, anotações), além de melhores práticas para destacar insights sem sobrecarregar o público.
    - Ferramentas sugeridas: `plotly`, `altair`, `bokeh`, `rich`, `python-pptx`

6. **Tipagem Estática em Python com Annotations**

- Discute como anotações de tipo (PEP 484) podem aumentar a robustez e legibilidade de projetos em larga escala. Mostra o uso de verificadores estáticos (mypy, pyright), integração com docstrings e frameworks de validação em tempo de execução para detectar erros antecipadamente e gerar documentação automática.
- Ferramentas sugeridas: `mypy`, `pyright`, `pydantic`, `dataclasses`

7. **Versionamento de Dados e Experimentos**

   - Examina princípios para rastrear versões de datasets e resultados de experimentos, garantindo reprodutibilidade e colaboração. Inclui modelos de snapshot, armazenamento de metadados (schema, descrições), ramificações de experimentos e integração genérica com sistemas de controle de versão, sem focar em uma ferramenta específica.
   - Ferramentas sugeridas: `dvc`, `git-lfs`, `lakeFS`, `mlflow`, `pachyderm`

8. **Tratamento de Erros & Logging em ETLs**

   - Ensina a estruturar blocos de tratamento de exceções (try/except) e a definir políticas de log (níveis, formatos estruturados) para pipelines de ingestão e transformação de dados. Cobre técnicas para capturar falhas em lote, gerar alertas e manter históricos de execução que facilitem a depuração e auditoria de processos.
   - Ferramentas sugeridas: `logging` (std-lib), `structlog`, `loguru`, `sentry-sdk`

9. **Lazy × Eager Evaluation em Pipelines de Transformação**

   - Explora os trade-offs entre avaliação preguiçosa (lazy) e ansiosa (eager) de coleções em Python. Apresenta geradores, iteradores e leitura chunk-wise para economizar memória e reduzir latência, comparando conceitualmente com execuções ansiosas em listas e DataFrames.
   - Ferramentas sugeridas: `itertools`, `more_itertools`, geradores `yield`, `toolz`, `dask`

10. **Paradigma Funcional Aplicado à Transformação de Dados**

    - Introduz técnicas funcionais para manipulação de coleções—uso de funções de ordem superior (map, filter, reduce), imutabilidade e composição de funções—que geram código mais declarativo e fácil de testar. Discute

quando adotar compreensões de lista, pipelines de transformações e como isso se alinha a padrões de análise reprodutível.
- Ferramentas sugeridas: `functools`, `toolz`, `fn.py`, list/dict comprehensions, `pyspark.sql.functions`

*Lista não exaustiva – grupos podem propor outras ferramentas correlatas (sujeitas à aprovação).

Observações Finais

- Evitem usar **NumPy, Pandas, Matplotlib ou Seaborn** como tema central (podem aparecer apenas em exemplos).
- Cada apresentação tem limite estrito de **10–15 min**; ensaio prévio é recomendado.
- Dúvidas sobre escopo ou profundidade devem ser enviadas **até 1 semana antes** da data de apresentação.