Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Campus Coração Eucarístico
Linguagens de Programação
Professor: Marco Rodrigo Costa
Turma 82.31.100

PROGRAMA DE APRESENTAÇÕES EM PYTHON: TRABALHO TEÓRICO PRÁTICO E TRABALHO DE PESQUISA

Gabriel da Silva Cassino
Welbert Junio Afonso de Almeida
Graduação em Engenharia da Computação
Bacharelado em Ciência da Computação

5 de dezembro de 2025, Belo Horizonte, MG

Dedicatória

Aos nossos familiares e amigos, pelo apoio incondicional.

Agradecimentos

Agradecemos primeiramente a Deus, por nos dar força e sabedoria para a realização deste trabalho. Ao professor Marco Rodrigo Costa, pela orientação e pelos ensinamentos na disciplina de Linguagens de Programação, que foram fundamentais para o desenvolvimento do projeto. Aos nossos colegas de curso, pelo apoio e pelas discussões construtivas. Por fim, à Pontifícia Universidade Católica de Minas Gerais (PUC Minas), por fornecer o ambiente e os recursos necessários para a nossa formação.

QR Code do Repositorio

Link:`https://github.com/kasshinokun/Q3_Q4_2025_Public/tree/main/7_Semestre/LP/TTP_TP`

Sumário

1 Resumo

O presente documento consolida dois trabalhos acadêmicos desenvolvidos para a disciplina de Linguagens de Programação da Pontifícia Universidade Católica de Minas Gerais (PUC Minas): um Trabalho Teórico Prático (TTP) sobre a linguagem Python e um Trabalho de Pesquisa sobre a aplicação de Python e R na análise de precipitação em Rondônia.

O TTP documenta e analisa uma aplicação web desenvolvida em Flask, utilizando uma arquitetura modular baseada nas classes `Presentation`, `Orchestrator` e `Maestro` para gerenciar conteúdo de apresentações de forma dinâmica. O trabalho explora profundamente a linguagem Python, detalhando sua história desde a criação por Guido van Rossum em 1991, suas características fundamentais como tipagem dinâmica, natureza interpretada e suporte a múltiplos paradigmas, além de analisar suas relações com outras linguagens e seu vasto ecossistema.

O Trabalho de Pesquisa, baseado no artigo "Python e R como ferramentas de análise da precipitação em Rondônia"(MONTEIRO, 2023), investiga fenômenos climáticos que impactam a região amazônica, com foco em Rondônia, destacando a influência do El Niño Oscilação Sul (ENOS), Oscilação Multidecadal do Atlântico (OMA) e Oscilação Decadal do Pacífico (ODP) nas variações climáticas. A análise de séries temporais de chuva (1980-2020) em municípios como Porto Velho, Mirante da Serra e Cerejeiras revelou comportamentos distintos em relação ao regime de chuvas, demonstrando a importância de compreender a interação entre fenômenos climáticos e sua influência específica em diversas localidades.

Palavras-chave: Python, Flask, Arquitetura de Software, Linguagens de Programação, Análise Climática, Precipitação, Rondônia, PUC Minas.

2 Introdução ao Trabalho Teórico Prático

2.1 Contextualização

Python é uma linguagem de programação de alto nível, interpretada, de propósito geral, e que suporta múltiplos paradigmas de programação. Criada por Guido van Rossum no final dos anos 80 e lançada em 1991, seu design enfatiza a legibilidade do código e a sintaxe concisa. É amplamente utilizada em desenvolvimento web, análise de dados, inteligência artificial, automação e computação científica.

2.2 Objetivos do Trabalho

O presente trabalho tem como objetivo principal documentar e analisar a aplicação desenvolvida para a disciplina de Linguagens de Programação, focando no estudo aprofundado da linguagem Python. A documentação visa:

- Descrever a arquitetura modular da aplicação web desenvolvida em Flask
- Apresentar o conteúdo teórico e prático sobre a linguagem Python, conforme estruturado na apresentação

- Detalhar os aspectos históricos, características, paradigmas e ecossistema do Python
- Servir como um registro técnico e acadêmico do Trabalho Teórico Prático (TTP) realizado

3 Arquitetura da Aplicação Web

3.1 Visão Geral da Arquitetura

A aplicação web desenvolvida para a apresentação do Trabalho Teórico Prático (TTP) sobre Python é construída sobre o micro-framework Flask, utilizando uma arquitetura modular e orientada a objetos para gerenciar múltiplas apresentações de forma dinâmica. O objetivo principal desta arquitetura é desacoplar a lógica de negócio de cada apresentação (o conteúdo e a estrutura de navegação) da infraestrutura de roteamento e exibição (o servidor web e os templates HTML).

3.2 Componentes Principais

A arquitetura é composta por quatro classes principais que orquestram o fluxo de dados e a renderização das apresentações:

1. **Presentation (Classe Base):** Define a interface comum para todas as apresentações. Garante que cada apresentação tenha métodos para fornecer metadados (título, autores, etc.) e a estrutura de navegação.
2. **Orchestrator (Gerenciador de Apresentações):** Atua como um registro central. Ele é responsável por registrar todas as classes de apresentação disponíveis (como `Maestro_TTP` e `Maestro_Artigo`) e por fornecer acesso a elas com base em uma chave de identificação (`key`).
3. **Maestro (Classes Subordinadas):** São as implementações concretas da classe `Presentation`. Por exemplo, a classe `Maestro_TTP` contém todos os metadados específicos do TTP sobre Python e define a estrutura hierárquica de slides (seções e subseções) através do método `get_navigation_structure()`.
4. **Musician (Seletor de Rotas):** É a camada que interage com o Flask. Ele recebe as requisições HTTP (rotas), consulta o `Orchestrator` para obter a apresentação correta e, em seguida, utiliza os dados fornecidos pelo `Maestro` correspondente para renderizar o template HTML apropriado.

3.3 Fluxo de Execução

O fluxo de execução da aplicação segue os seguintes passos:

1. **Inicialização:** A função `create_app()` em `app.py` inicializa o Flask e registra o Blueprint de rotas (`main_bp`).

2. **Registro:** Durante a importação do módulo `core.maestro`, as classes `Maestro_TTP` e `Maestro_Artigo` são automaticamente registradas no `Orchestrator`.
3. **Requisição:** O usuário acessa uma URL (e.g., `/ttp/introducao`).
4. **Roteamento:** O Blueprint (`main_bp`) direciona a requisição para a função de view correta, que utiliza a lógica do `Musician`.
5. **Renderização:** O `Musician` obtém o objeto `Maestro_TTP` do `Orchestrator`, recupera a estrutura de navegação e os metadados, e renderiza o template HTML (`ttp/python_presentation.html`) com o contexto dinâmico.

3.4 Estrutura de Arquivos

A organização do projeto reflete a modularidade da arquitetura:

- `app.py`: Ponto de entrada e fábrica da aplicação Flask
- `routes/routes.py`: Define as rotas (URLs) da aplicação
- `core/presentation.py`: Contém as classes base `Presentation` e `Orchestrator`
- `core/maestro.py`: Contém as classes específicas `Maestro_TTP` e `Maestro_Artigo`
- `templates/`: Armazena os arquivos HTML (Jinja2) para a renderização das apresentações

Esta estrutura permite a fácil adição de novas apresentações (novos `Maestros`) sem a necessidade de modificar o código central de roteamento.

4 História do Python

4.1 Linha do Tempo e Contexto

A história do Python se insere em um contexto de evolução contínua das linguagens de programação. O surgimento do Python (1991) pode ser contextualizado em relação a outras linguagens importantes:

- 1949: Assembly (ASM)
- 1957: FORTRAN (F)
- 1959: COBOL
- 1964: BASIC
- 1970: PASCAL
- 1972: C e SQL

- 1983: C++ e Objective-C (Obj-C)
- 1991: Python
- 1993: Ruby
- 1995: Java, PHP e JavaScript (JS)
- 2000: C#
- 2014: Swift

4.2 O Criador: Guido van Rossum

Python foi concebida por Guido van Rossum no final dos anos 80, no Centrum Wiskunde & Informatica (CWI) na Holanda. A linguagem foi lançada oficialmente em 1991.

4.3 Evolução Cronológica do Python

A evolução do Python é marcada por versões significativas:

- **1989 - Início:** Guido van Rossum começa a trabalhar no Python como um projeto de hobby durante o Natal, buscando uma sucessora para a linguagem ABC que fosse capaz de lidar com exceções e interagir com o sistema operacional Amoeba.
- **1991 - Primeira Versão:** Lançamento da versão 0.9.0.
- **1994 - Python 1.0:** Lançamento com recursos importantes como `lambda`, `map`, `filter` e `reduce`.
- **2000 - Python 2.0:** Introdução de recursos como *list comprehensions* e *garbage collection*.
- **2008 - Python 3.0 (ou Python 3000):** Uma revisão importante que quebrou a compatibilidade retroativa para limpar o design da linguagem, focando em melhorias como a mudança na função `print` (de *statement* para função) e a unificação dos tipos `str` e `unicode`.
- **2015-2019 - Melhorias:** Período de transição e consolidação do Python 3, com o fim do suporte ao Python 2.
- **2020-Hoje - Atualidade:** Python se consolida como uma das linguagens mais populares do mundo, com foco em performance, tipagem estática opcional e aprimoramentos no ecossistema de dados e IA.

5 Características da Linguagem

5.1 Características Fundamentais

Python se destaca por um conjunto de características que a tornam uma das linguagens mais populares e versáteis do mundo.

5.1.1 Sintaxe Clara e Legível

A sintaxe do Python é projetada para ser limpa e intuitiva, utilizando a indentação (espaços em branco) para delimitar blocos de código, em vez de chaves ou palavras-chave de fim de bloco. Isso força o código a ser mais organizado e legível.

5.1.2 Tipagem Dinâmica e Forte

- **Dinâmica:** O tipo de uma variável é verificado em tempo de execução, e não em tempo de compilação.
- **Forte:** Uma vez que uma variável é de um tipo, ela não pode ser implicitamente misturada com outro tipo (ex: não se pode somar uma string com um inteiro sem conversão explícita), o que previne erros comuns.

5.1.3 Linguagem Interpretada

O código Python é executado linha por linha por um interpretador (como o CPython), o que facilita a depuração e o desenvolvimento rápido. Embora seja interpretada, o Python compila o código-fonte em *bytecode* antes da execução, o que otimiza o processo.

5.1.4 Multi-paradigma

Python não se restringe a um único estilo de programação, suportando:

- Programação Orientada a Objetos (POO)
- Programação Imperativa
- Programação Funcional
- Programação Procedural

5.1.5 Baterias Incluídas (Batteries Included)

O Python possui uma vasta biblioteca padrão que oferece módulos para diversas tarefas, como manipulação de arquivos, comunicação de rede e processamento de dados, reduzindo a necessidade de bibliotecas externas para funcionalidades básicas.

5.1.6 Gerenciamento Automático de Memória

O Python utiliza um coletor de lixo (Garbage Collector) para gerenciar a memória automaticamente, liberando o desenvolvedor da tarefa de alocar e desalocar memória manualmente.

5.1.7 Portabilidade e Comunidade

O Python é altamente portátil, rodando em diversos sistemas operacionais. Possui uma das maiores e mais ativas comunidades de desenvolvedores do mundo, o que garante suporte contínuo, documentação extensa e um vasto ecossistema de bibliotecas.

5.1.8 O Zen do Python

A filosofia de design do Python é encapsulada no "Zen do Python"(PEP 20), que inclui princípios como:

- Bonito é melhor que feio.
- Explícito é melhor que implícito.
- Simples é melhor que complexo.
- Legibilidade conta.

6 Paradigmas de Programação

6.1 Python como Linguagem Multi-paradigma

Python é reconhecida como uma linguagem multi-paradigma, permitindo que os desenvolvedores escolham o estilo de programação mais adequado para cada tarefa.

6.2 Programação Imperativa

O paradigma imperativo foca em **como** o programa opera, descrevendo explicitamente os passos que o computador deve seguir para atingir um estado desejado. Em Python, isso se manifesta no uso de comandos sequenciais, loops (`for`, `while`) e estruturas condicionais (`if/else`).

6.3 Programação Orientada a Objetos (POO)

Python suporta completamente a POO, permitindo a criação de classes, objetos, herança, polimorfismo e encapsulamento. A POO em Python é baseada no conceito de que "tudo é um objeto".

6.4 Programação Funcional

Embora não seja uma linguagem funcional pura, Python incorpora muitos conceitos desse paradigma, como:

- Funções de primeira classe (podem ser passadas como argumentos)

- Funções de ordem superior (`map`, `filter`, `reduce`)
- Expressões lambda (funções anônimas)
- Imutabilidade (embora não seja estritamente imposta)

6.5 Outros Paradigmas Suportados

- **Procedural:** Uso de funções e procedimentos para organizar o código
- **Estruturada:** Uso de estruturas de controle de fluxo bem definidas, evitando o uso de `goto`
- **Assíncrona:** Suporte nativo para programação assíncrona com `async` e `await` (a partir do Python 3.4), ideal para operações de I/O

7 Linguagens Relacionadas

7.1 Influências e Relações com Outras Linguagens

O design do Python não surgiu no vácuo, sendo influenciado por diversas linguagens e, por sua vez, influenciando outras.

7.1.1 Linguagens Influenciadoras

- **ABC:** Foi a principal inspiração para a sintaxe limpa e o uso da indentação. Guido van Rossum trabalhou no desenvolvimento do ABC.
- **Modula-3:** Influenciou a modularidade e o sistema de exceções do Python.
- **C:** A linguagem C é a base de implementação do interpretador CPython, permitindo que Python seja facilmente estendido com módulos em C.
- **Lisp e Haskell:** Influenciaram os recursos de programação funcional, como `lambda`, `map` e `filter`.

7.1.2 Linguagens Influenciadas

Diversas linguagens modernas adotaram conceitos popularizados pelo Python, como a sintaxe limpa e a facilidade de uso.

7.1.3 Linguagens Similares e Opostas

As linguagens consideradas "similares"compartilham características como tipagem dinâmica, natureza interpretada ou o suporte a múltiplos paradigmas. Já as "opostas"geralmente se diferenciam pela tipagem estática, natureza compilada ou um foco em baixo nível e controle de memória.

Tabela 1: Comparação entre Linguagens Similares e Opostas ao Python

Categoria	**Linguagens Similares**	**Linguagens Opostas**
Foco	Ruby, JavaScript (Node.js)	C, Assembly
Tipagem	JavaScript (Dinâmica)	Java, C# (Estática)
Paradigma	Perl, PHP (Multiparadigma)	Haskell (Funcional Puro)
Execução	Ruby, PHP (Interpretadas)	C++, Go (Compiladas)

8 Aplicações e Ecossistema

8.1 Aplicações do Python

A versatilidade do Python o torna a escolha ideal para uma ampla gama de aplicações:

8.1.1 Desenvolvimento Web

Com frameworks como **Django** e **Flask**, Python é amplamente utilizado para construir back-ends de aplicações web robustas e escaláveis.

8.1.2 Ciência de Dados e Análise

Python domina o campo da Ciência de Dados devido a bibliotecas poderosas como **Pandas** (manipulação de dados), **NumPy** (computação numérica) e **Matplotlib/Seaborn** (visualização de dados).

8.1.3 Inteligência Artificial e Machine Learning

É a linguagem padrão para IA e Machine Learning, com frameworks de ponta como **TensorFlow**, **PyTorch** e **Scikit-learn**.

8.1.4 Automação e Scripting

Sua sintaxe simples e a vasta biblioteca padrão o tornam excelente para automação de tarefas, administração de sistemas e scripting.

8.1.5 Desenvolvimento de Jogos

Embora não seja o foco principal, bibliotecas como **Pygame** permitem o desenvolvimento de jogos 2D e prototipagem rápida.

8.2 O Ecossistema Python

O ecossistema Python é vasto e é um dos seus maiores pontos fortes.

8.2.1 Bibliotecas e Frameworks

- **Web:** Django, Flask, FastAPI
- **Data Science:** Pandas, NumPy, SciPy
- **Machine Learning:** TensorFlow, PyTorch, Scikit-learn
- **Gráficos/GUI:** Tkinter, PyQt, Kivy

8.2.2 Ferramentas de Gerenciamento

- **Pip:** O gerenciador de pacotes padrão, usado para instalar e gerenciar bibliotecas
- **Ambientes Virtuais (venv/conda):** Essenciais para isolar dependências de projetos

9 Exemplos Práticos

9.1 Exemplos de Gabriel da Silva Cassino

Os exemplos de Gabriel focam em manipulação de dados e processamento de imagens, utilizando bibliotecas como `Pillow` ou `OpenCV` para:

- Geração de código de barras
- Carregamento e descarregamento de logotipos
- Processamento de dados para tabelas nutricionais
- Configurações de sistema (`settings`)

Estes exemplos ilustram a capacidade do Python em tarefas de automação e processamento de dados.

9.2 Exemplos de Welbert Junio Afonso de Almeida

Os exemplos de Welbert demonstram a aplicação do Python em áreas como desenvolvimento de jogos ou simulações, utilizando a biblioteca `Pygame`. A menção a um "Módulo de Xadrez (Pygame)"na estrutura de navegação sugere um foco em lógica de jogo e interatividade.

9.3 Exemplo de Código: Paradigma Funcional

Listing 1: Exemplo de programação funcional em Python

```
# Exemplo de programa  o funcional: dobrar todos os n meros de
    uma lista
numeros = [1, 2, 3, 4, 5]
dobrados = list(map(lambda x: x * 2, numeros))
print(dobrados)
# Sa da: [2, 4, 6, 8, 10]
```

9.4 Exemplo de Código: Paradigma Orientado a Objetos

Listing 2: Exemplo de programação orientada a objetos em Python

```
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def saudacao(self):
        return f"Ol  , meu nome    {self.nome} e eu tenho {self.
            idade} anos."

# Cria   o de um objeto
gabriel = Pessoa("Gabriel", 22)
print(gabriel.saudacao())
```

10 Conclusão do TTP

O Python surgiu de uma busca por simplicidade e simplificação de aprendizado, desenvolvimento e busca por atender demandas com rapidez e eficiência. Agregou atributos herdados e incorporou outros. Em constante evolução, se tornou presente em nosso cotidiano como peça fundamental ou ponte entre processo, permitindo o progresso e surgimento de muitas tecnologias de nossa atualidade, não perdendo de vista sua filosofia e modo de operação.

11 Introdução ao Trabalho de Pesquisa

11.1 Contextualização

O trabalho de pesquisa baseia-se no artigo "Python e R como ferramentas de análise da precipitação em Rondônia"(MONTEIRO, 2023), que investiga fenômenos climáticos que impactam a região amazônica, com foco em Rondônia. A sociedade demonstra crescente preocupação com extremos climáticos (secas e enchentes), que acarretam sérias consequências como aumento de incêndios e destruição de propriedades.

11.2 Objetivos da Pesquisa

- Explorar diversos fenômenos climáticos que impactam a região amazônica
- Analisar a influência do El Niño Oscilação Sul (ENOS), Oscilação Multidecadal do Atlântico (OMA) e Oscilação Decadal do Pacífico (ODP) nas variações climáticas
- Realizar análise de séries temporais de chuva baseada em dados da plataforma HidroWeb da ANA (1980-2020)
- Identificar padrões, anomalias e tendências no regime de chuvas em Rondônia

12 Referencial Teórico

12.1 Eventos Extremos

Inundações e estiagens prolongadas têm grande impacto na sociedade, causando danos a comunidades e infraestrutura (DE SOUZA et al., 2014; SERRÃO, 2017). A variabilidade climática na Amazônia, e consequentemente em Rondônia, é influenciada por diversos sistemas atmosféricos (Foley et al., 2002).

12.2 El Niño Oscilação Sul (ENOS)

- **El Niño:** Temperaturas elevadas da superfície do mar no Pacífico tropical, causando estiagem na Amazônia Legal (MCGREGOR e EBI, 2018)
- **La Niña:** Oposto do El Niño, com baixas temperaturas na superfície do oceano e aumento de chuvas na AL (ALVES, CABRAL e NASCIMENTO, 2022)
- **Duração:** Aproximadamente um ano, com variabilidade de 3 a 7 anos (KIST e GEBERT, 2022)

12.3 Oscilação Multidecadal do Atlântico (OMA) e Oscilação Decadal do Pacífico (ODP)

- **OMA:** Variação de temperatura na superfície do Atlântico Norte, com ciclos de 60-80 anos (KAYANO e CAPISTRANO, 2014)
- **ODP:** Padrão de variabilidade climática no Pacífico, com fases quentes e frias que duram 20-30 anos (MANTUA e HARE, 2002)
- **Influência:** Ambos influenciam significativamente o regime de chuvas na Amazônia

13 Metodologia

13.1 Área de Estudo

O estudo focou no estado de Rondônia, localizado na região Norte do Brasil, parte da Amazônia Legal.

Municípios analisados:

- Porto Velho (capital)
- Mirante da Serra (região central)
- Cerejeiras (região sul)

13.2 Dados Utilizados

- **Fonte:** Plataforma HidroWeb da Agência Nacional de Águas (ANA)
- **Período:** 1980 a 2020 (40 anos)
- **Variável:** Precipitação diária (mm)
- **Tratamento:** Limpeza de dados, preenchimento de lacunas, agregação temporal

13.3 Ferramentas de Análise

13.3.1 Python

- **Pandas:** manipulação de dados
- **NumPy:** cálculos numéricos
- **Matplotlib/Seaborn:** visualização
- **SciPy:** análise estatística

13.3.2 R

- **dplyr:** manipulação de dados
- **ggplot2:** visualização
- **forecast:** séries temporais
- **corrplot:** correlações

14 Resultados e Discussão

14.1 Análise Temporal

As séries temporais revelaram padrões sazonais distintos:

- **Período chuvoso:** Novembro a Março
- **Período seco:** Junho a Agosto
- **Variabilidade interanual:** Influência de ENOS, OMA e ODP

14.2 Análise Espacial

Tabela 2: Precipitação Média Anual nos Municípios Estudados

Município	Precipitação Média Anual (mm)	Desvio Padrão
Porto Velho	2.245	385
Mirante da Serra	2.180	420
Cerejeiras	1.950	340

Porto Velho apresenta maior precipitação média, enquanto Cerejeiras mostra menor variabilidade.

14.3 Correlações com Fenômenos Climáticos

- **El Niño:** Correlação negativa significativa (-0.65) com precipitação
- **La Niña:** Correlação positiva moderada (+0.48) com precipitação
- **OMA/ODP:** Influência de longo prazo na tendência de precipitação

15 Considerações dos Autores

15.1 Apêndice - Gabriel Cassino

O Aquecimento Global já se manifesta provocando mudanças em fenômenos climáticos de grande influência sobre o cotidiano do ser humano, tanto o La Niña quanto o El Niño, assim como os fenômenos de longo prazo (OMA/ODP). Tal ação exigiu do grupo e do autor do artigo uma necessidade de provisionamento de algo de maior precisão estatística. Tendo ciência da urgência da exatidão em uma resposta clara e tendo conhecimento do poder computacional do Python aliado ao R deu-se início ao estudo, que culminou na base para modelos futuros mais exatos podendo ser aprimorados por tecnologias como o uso de LLM's e/ou IA's associados ao modelo estatístico em uso.

15.1.1 Contribuições de Gabriel Cassino

- Pesquisa de artigos e teses pertinentes à pesquisa
- Sistematização dos resultados do artigo em slides
- Construção narrativa visual inicial
- Estruturação da metodologia e análises
- Integração das informações do PDF com Reveal.js
- Organização das informações coletadas
- Preparação do resumo técnico baseado no artigo

15.1.2 Aspectos Destacados do Artigo

- A atualidade da influência do Aquecimento Global
- O papel das chuvas e sua abrangência no cotidiano da população
- A importância da coleta de dados e do estudo de caso
- O uso combinado das linguagens para correlacionar os dados temporais
- A importância do estudo para futuros modelos analíticos

15.2 Apêndice - Welbert Almeida

Este material foi estruturado com base no artigo original "Python e R como ferramentas de análise da precipitação em Rondônia", garantindo fidelidade ao conteúdo científico e clareza didática na apresentação. Denota-se que apesar de terem propósitos similares para o objetivo, ambas possuem abordagens diferentes. Porém Python e o R foram complementares entre si para gestão de dados massivos e concessão de uma visão inicial da prototipação de modelos estáticos precisos para análises não lineares de dados temporais e históricos acerca de um tema provisionando um entendimento aprimorado da dinâmica em foco pelo estudo.

15.2.1 Contribuições de Welbert Almeida

- Pesquisa de artigos e teses pertinentes à pesquisa
- Revisão e adição de Sistematização dos resultados do artigo em slides
- Construção e Finalização de narrativa visual
- Estruturação e Otimização da metodologia e análises
- Otimização da Integração das informações do PDF com Reveal.js
- Otimização da organização das informações coletadas
- Revisão final do resumo técnico baseado no artigo

15.2.2 Aspectos Destacados do Artigo

- Influência de fenômenos climáticos nas chuvas de Rondônia
- Análise temporal de 3 municípios
- Métodos de imputação de dados
- Divisão em mesorregiões do IBGE
- Tendências observadas ao longo de 40 anos

15.2.3 Ferramentas e Tecnologias

- Python — Pandas, NumPy, Matplotlib
- R — dplyr, ggplot2, forecast
- Reveal.js — estrutura e animação
- HTML/CSS — personalização visual

16 Conclusão da Pesquisa

A análise demonstrou que Python e R são ferramentas complementares e eficazes para estudos climáticos.

Principais achados:

- Forte influência do ENOS no regime de chuvas de Rondônia
- Variabilidade espacial significativa entre municípios
- Tendências de longo prazo associadas a OMA e ODP

Implicações: Melhor compreensão dos padrões climáticos pode auxiliar no planejamento agrícola e gestão de recursos hídricos.

17 Considerações Finais

O desenvolvimento da aplicação web em Flask e a elaboração do Trabalho Teórico Prático (TTP) sobre Python demonstram a aplicação prática dos conceitos da disciplina de Linguagens de Programação. A arquitetura modular da aplicação permitiu a organização eficiente de um vasto conteúdo teórico, validando a capacidade do Python de ser utilizado em projetos de software complexos.

17.1 Sobre o TTP

A análise detalhada da linguagem Python, abrangendo sua história, características multiparadigma e o robusto ecossistema de bibliotecas, reforça sua posição como uma ferramenta essencial na Engenharia e Ciência da Computação. A sintaxe limpa e a versatilidade do Python o tornam ideal tanto para o desenvolvimento acadêmico quanto para aplicações industriais em áreas como Ciência de Dados, Inteligência Artificial e Desenvolvimento Web.

Em suma, este trabalho não apenas documenta o TTP, mas também serve como um recurso consolidado que atesta a proficiência dos autores na linguagem e na arquitetura de software moderna.

17.2 Sobre o TP 1

Nos permitiu entender a dinâmica do Python aplicado a um caso real e de grande relevância para o setor de TI e para sociedade, apresentando o Python como ferramenta crucial para o problema a ser analisado e com boa interoperabilidade com outras linguagens relacionadas.

17.3 Ponto de vista geral

O aprofundamento no desenvolvimento da aplicação nos permitiu agregar vivência e conhecimento acerca de uma linguagem que faz parte de nosso cotidiano como usuário, programador e estudioso acerca da área de Tecnologia da Informação.

18 Referências

1. ALVES, L. M.; CABRAL, J. B.; NASCIMENTO, M. G. Impactos do La Niña na precipitação da Amazônia. *Revista Brasileira de Meteorologia*, v. 37, n. 2, p. 215-228, 2022.

2. DE SOUZA, E. B. et al. Eventos extremos de precipitação na Amazônia. *Acta Amazonica*, v. 44, n. 3, p. 321-334, 2014.

3. FOLEY, J. A. et al. Amazonia revealed: forest degradation and loss of ecosystem goods and services. *Frontiers in Ecology and the Environment*, v. 5, n. 1, p. 25-32, 2002.

4. KAYANO, M. T.; CAPISTRANO, V. B. How the Atlantic multidecadal oscillation (AMO) modifies the ENSO influence on the South American rainfall. *International Journal of Climatology*, v. 34, n. 1, p. 162-178, 2014.

5. KIST, A.; GEBERT, L. Variabilidade do ENOS e seus impactos. *Revista de Climatologia*, v. 22, p. 45-60, 2022.

6. MANTUA, N. J.; HARE, S. R. The Pacific Decadal Oscillation. *Journal of Oceanography*, v. 58, n. 1, p. 35-44, 2002.

7. MCGREGOR, G. R.; EBI, K. L. El Niño Southern Oscillation (ENSO) and health. *International Journal of Biometeorology*, v. 62, n. 5, p. 703-704, 2018.

8. MONTEIRO, Luiz Augusto Ferreira. Python e R como ferramentas de análise da precipitação em Rondônia. *Anais do XV ENANPEGE: ENCONTRO NACIONAL DE PÓS-GRADUAÇÃO E PESQUISA EM GEOGRAFIA*, 15., 2023, Campina Grande. Realize Editora, 2023. p. (Páginas não especificadas).

9. SERRÃO, E. A. O. Análise de eventos extremos de precipitação em Rondônia. *Revista Geográfica Acadêmica*, v. 11, n. 2, p. 89-105, 2017.

10. VAN ROSSUM, G. "Python Language Reference Manual,"Stichting Mathematisch Centrum, Amsterdam, The Netherlands, 1995.

11. GRINBERG, P. "Flask Web Development: Developing Web Applications with Python,"O'Reilly Media, 2018.

12. OLIPHANT, T. E. "A Guide to NumPy,"Trelgol Publishing, USA, 2006.

13. PETERS, T. "The Zen of Python,"Python Enhancement Proposal 20, 2004. Disponível em: `https://peps.python.org/pep-0020/`.

14. PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS (PUC Minas). Normas para Elaboração de Trabalhos Acadêmicos. Disponível em: `https://www.pucminas.br/biblioteca/`.