

```
// Filter and map an array of numbers

// This function processes an array of numbers.
// It first filters out the odd numbers and then squares the even numbers.

function processNumbers(numbers) {
  // Filter the array to keep only even numbers
  const evenNumbers = numbers.filter(n => n % 2 === 0);

  // Map the even numbers to their squares
  const squaredEvenNumbers = evenNumbers.map(n => n * n);

  // Return the new array with squared even numbers
  return squaredEvenNumbers;
}





// Example usage:
// An array of numbers to process
const numbers = [1, 2, 3, 4, 5, 6];

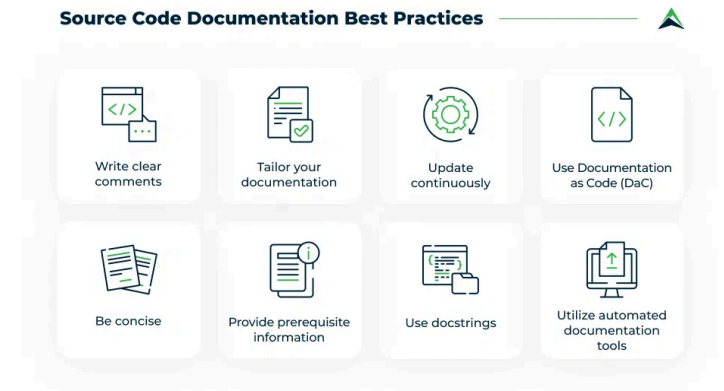
// Process the array and log the result
console.log(processNumbers(numbers)); // [4, 16, 36]
```

Documentação Técnica, Versionamento e Boas Práticas

Aula 10 - UC2: Ciência de Dados

Objetivos da Aula

-  Compreender a **importância da documentação técnica** e seu papel na sustentabilidade de projetos
-  Aprender a criar **documentação eficiente** para scripts e funções usando comentários, docstrings e READMEs
-  Entender os **fundamentos de versionamento** com Git e GitHub para controle e colaboração
-  Desenvolver **boas práticas de documentação** e organização de código como parte da ética profissional



Abertura e Contexto

"O código mais bonito é aquele que não precisa de explicação, mas o mais útil é aquele que a tem."

A documentação é muito mais que uma formalidade técnica: é a **ponte essencial** que conecta diferentes aspectos de um projeto de dados.

A Documentação como Ponte entre:

 **O Código** - Transformando linhas de código em conhecimento acessível

 **O Analista** - Atual e futuro, permitindo continuidade e evolução

 **O Cliente** - Traduzindo soluções técnicas em valor de negócio

```
// Filter and map an array of numbers

// This function processes an array of numbers.
// It first filters out the odd numbers and then squares the even numbers.

function processNumbers(numbers) {
  // Filter the array to keep only even numbers
  const evenNumbers = numbers.filter(n => n % 2 === 0);

  // Map the even numbers to their squares
  const squaredEvenNumbers = evenNumbers.map(n => n * n);

  // Return the new array with squared even numbers
  return squaredEvenNumbers;
}

// Example usage:
// An array of numbers to process
const numbers = [1, 2, 3, 4, 5, 6];

// Process the array and log the result
console.log(processNumbers(numbers)); // [4, 16, 36]
```

Por que Documentar?



Comunicação

Permite que outros membros da equipe entendam o seu trabalho rapidamente, facilitando a colaboração e reduzindo o tempo de integração de novos desenvolvedores.



Sustentabilidade

Garante que você (ou outra pessoa) consiga retomar o projeto meses depois, entendendo as decisões tomadas e o funcionamento do código sem precisar reaprender tudo.



Validação

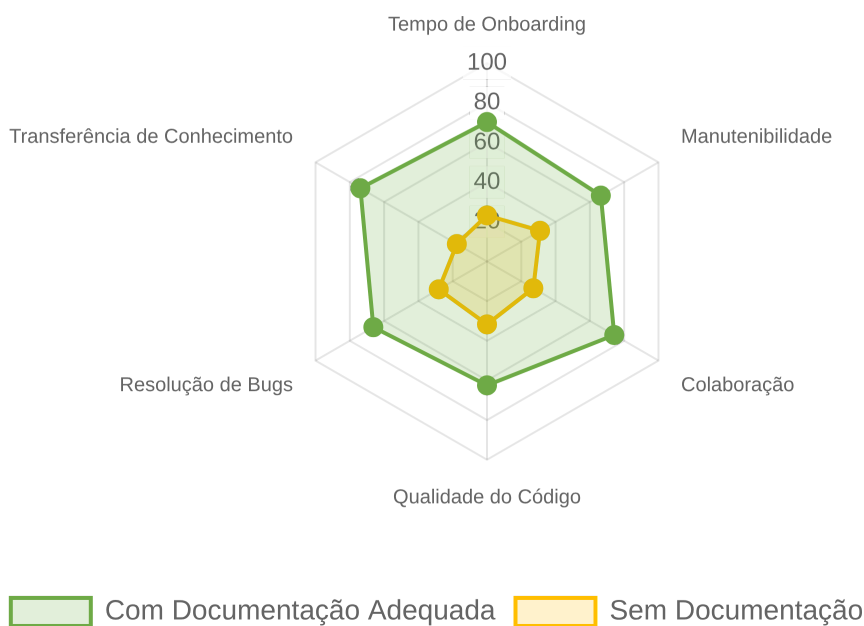
Explica a lógica por trás de decisões complexas de limpeza ou modelagem de dados, permitindo que outros validem sua abordagem e metodologia.



Ética

Documentar é um ato de responsabilidade e disciplina profissional, demonstrando compromisso com a qualidade e transparência do trabalho realizado.

Impacto da Documentação no Ciclo de Vida do Projeto



Tipos de Documentação

Comentários Inline

Notas curtas dentro do código que explicam trechos específicos ou decisões pontuais. São úteis para esclarecer lógicas complexas ou não óbvias.

Comentário Inline

```
# Filtra outliers usando o método do desvio absoluto
def filtrar_outliers(dados, limite=2.0):
    # Calcula a média dos dados
    media = sum(dados) / len(dados)

    # Calcula os desvios absolutos
    desvios = [abs(x - media) for x in dados]
```

” Docstrings

Descrições detalhadas de funções, métodos e classes, incluindo parâmetros, retornos e exemplos de uso. São essenciais para documentação de API.

Docstring

```
def calcular_estatisticas(dados):
    """
    Calcula estatísticas básicas de um conjunto de dados.

    Args:
        dados (list): Lista de valores numéricos

    Returns:
        tuple: (média, mediana, moda)
    """
```

📄 README

Documento de "boas-vindas" do projeto que explica seu propósito, como executá-lo e quais são os resultados esperados. É a primeira documentação consultada.

README (Markdown)

```
# Análise Estatística

Uma biblioteca para análise estatística básica

## Instalação
```
pip install analise-estatistica
```
```

Atividade Prática 1 - Análise de Documentação Exemplar

Exemplos de Código com Diferentes Níveis de Documentação

Exemplo C: Documentação Exemplar

```
def calcular_estatisticas(dados):  
    if not dados:  
        return None, None, None  
  
    soma = sum(dados)  
    tamanho = len(dados)  
    media = soma / tamanho  
  
    dados_ordenados = sorted(dados)  
    if tamanho % 2 == 0:  
        mediana = (dados_ordenados[tamanho // 2 - 1] + dados_ordenados[tamanho // 2]) / 2  
    else:  
        mediana = dados_ordenados[tamanho // 2]  
  
    contagem = {}  
    for valor in dados:  
        if valor in contagem:  
            contagem[valor] += 1  
        else:  
            contagem[valor] = 1
```

Tarefa em Grupo

- 1 Analisar a clareza, organização e suficiência da documentação do Exemplo C
- 2 Identificar os padrões que tornam o código compreensível (separação de blocos lógicos, uso de linhas em branco)
- 3 Comparar os três exemplos e discutir as diferenças de legibilidade e manutenibilidade
- 4 Apresentar conclusões sobre a importância da documentação adequada

Dicas para Análise

- Observe como a docstring explica o propósito da função
- Verifique se os parâmetros e retornos estão bem documentados
- Avalie se os comentários explicam o "porquê" e não apenas o "o quê"

A Arte da Docstring

Estrutura da Docstring (Estilo Google)

Descrição Breve

Uma frase concisa na primeira linha que resume o propósito da função ou classe.

Sessão Args

Lista todos os parâmetros com seus tipos e descrições.

nome_param (tipo): Descrição

Sessão Returns

Descreve o que a função retorna, incluindo tipo e significado.

tipo: Descrição do retorno

Sessão Raises

Lista as exceções que podem ser lançadas e em quais condições.

TipoExcecao: Quando ocorre

Dica Profissional

Mantenha suas docstrings atualizadas! Uma documentação desatualizada é pior que nenhuma documentação, pois pode levar a erros e confusão.

Exemplo de Docstring Completa

Função com Docstring no Estilo Google

```
def calcular_estatisticas(dados):  
    """  
    Calcula a média, mediana e moda de um conjunto de dados numéricos.  
  
    Esta função analisa um conjunto de dados e retorna estatísticas descritivas básicas. Se vazio, retorna None para todas.  
  
    Args:  
        dados (list): Lista de valores numéricos para análise.
```

Visualização da Docstring no Help

```
# Resultado ao executar help(calcular_estatisticas)
```

```
Help on function calcular_estatisticas:
```

```
calcular_estatisticas(dados)  
    Calcula a média, mediana e moda de um conjunto de dados numéricos.
```

```
    Esta função analisa um conjunto de dados e retorna estatísticas descritivas básicas. Se vazio, retorna None para todas.
```

Atividade Prática 2 - Documentação de Scripts

☰ Instruções da Atividade

1 Seleccione um Script

Use um script Python/Colab de aulas anteriores

2 Adicione Comentários

Documente passos cruciais da análise

3 Crie Docstrings

Para cada função, adicione Args e Returns

4 Elabore um README

Explique objetivo e execução do script

💡 Dicas para Documentação Eficiente

- Documente o **porquê**, não só o **como**
- Use linguagem clara e direta
- Inclua exemplos para funções complexas

</> Exemplos de Documentação

Comentários Inline

```
# Removendo valores ausentes da coluna 'renda'
df = df.dropna(subset=['renda'])

# Convertendo valores de string para float
df['renda'] = df['renda'].str.replace(',', '.').astype(float)
```

Exemplo de Docstring

```
def calcular_estatisticas(dados):
    """
    Calcula estatísticas básicas de um conjunto de dados.

    Args:
        dados (list): Lista de valores numéricos para análise.

    Returns:
        tuple: Uma tupla contendo (média, mediana, desvio padrão).
    """
    media = sum(dados) / len(dados)
    dados_ordenados = sorted(dados)
```

Exemplo de README (Markdown)

```
# Análise de Dados de Vendas

Este notebook realiza a limpeza e análise exploratória dos dados de vendas do primeiro trimestre de 2025.

## Como executar

1. Execute todas as células em ordem
```

Versionamento e GitHub

Atividade Prática 3 - Uso Prático de Git

Passo a Passo

1 Inicialização

Crie um novo repositório Git para o projeto

```
$ git init
```

2 Adição de Arquivos

Adicione os arquivos do projeto ao staging

```
$ git add .
```

3 Commit

Salve as mudanças com uma mensagem clara

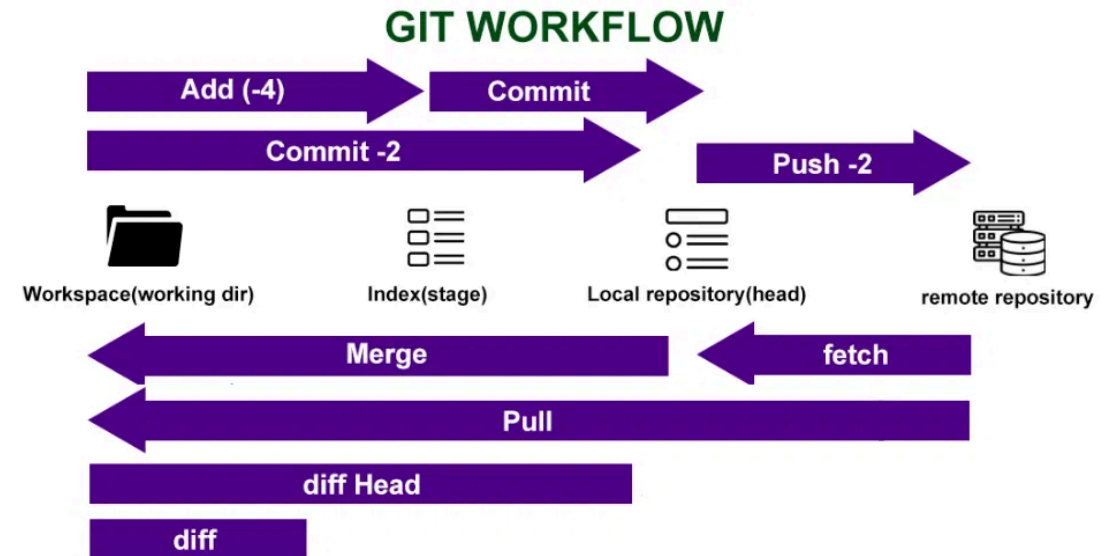
```
$ git commit -m "feat: implementa função de análise estatística"
```

4 Simulação de Colaboração

Simule um cenário onde um colega adiciona documentação

```
$ git pull  
$ git push
```

Fluxo de Trabalho Git



✓ Regras para Mensagens de Commit

- **Seja específico:** A mensagem deve descrever claramente o que foi alterado
Ex: "fix: corrige cálculo de média ponderada"
- **Use prefixos padronizados:** feat (nova funcionalidade), fix (correção), docs (documentação), refactor (refatoração), etc.
- **Mantenha conciso:** Primeira linha com no máximo 72 caracteres
- **Use o imperativo:** "Adiciona função" em vez de "Adicionada função"

Atividade Prática 4 - Guia de Boas Práticas

Instruções da Atividade

Objetivo

Elaborar um guia de boas práticas para documentação e organização de código que possa ser utilizado como referência pela equipe.

Procedimento

- 1 Forme grupos de 3-4 pessoas para colaboração
- 2 Defina categorias para o guia (ex: comentários, docstrings, commits, README)
- 3 Para cada categoria, estabeleça regras claras e exemplos de boas e más práticas
- 4 Crie um documento Markdown com o guia completo
- 5 Apresente o guia para a turma, destacando os pontos mais importantes

Categorias Sugeridas

- Regras para Comentários
- Padrão de Nomes de Variáveis
- Regras para Mensagens de Commit
- Conteúdo Mínimo do README
- Estrutura de Docstrings
- Organização de Arquivos

Exemplos para o Guia

Exemplo de Regra para Comentários

✓ **BOM:** Comentar o "porquê", não apenas o "como"

```
# Usando desvio absoluto em vez de desvio padrão para
# reduzir a influência de outliers extremos
desvios = [abs(x - media) for x in dados]
```

✗ **RUIM:** Comentários óbvios ou redundantes

```
# Calcula a média
media = sum(dados) / len(dados)
```

Exemplo de Regra para Mensagens de Commit

✓ **BOM:**
feat: implementa função de normalização de dados
fix: corrige cálculo da mediana para arrays vazios
docs: atualiza documentação da função calcular_estatisticas

✗ **RUIM:**
fix bug
update
changes

Exemplo de Estrutura de README

Seções Obrigatórias:

- Título e Descrição
- Instalação
- Uso
- Contribuição
- Licença

Seções Recomendadas:

- Badges (status do build, cobertura de testes)
- Requisitos
- Exemplos
- FAQ

Ética Profissional na Documentação

Encerramento e Próximos Passos

✓ O Que Aprendemos

Conceitos-Chave

- 📄 A importância da documentação técnica como ponte entre código, analista e cliente
- ” Estrutura e componentes de docstrings eficientes no estilo Google
- 🔑 Fundamentos de versionamento com Git e GitHub para controle e colaboração
- ✓ Boas práticas para mensagens de commit e organização de código

"Documentação e versionamento não são apenas boas práticas técnicas, são compromissos éticos com a qualidade e sustentabilidade do seu trabalho."

➔ Próximos Passos



Integração com o Projeto Integrador

Aplique os conceitos de documentação e versionamento no seu projeto integrador, criando um repositório Git organizado e bem documentado.



Colaboração em Equipe

Pratique o fluxo de trabalho colaborativo com Git, realizando pull requests e revisões de código com seus colegas de equipe.



Documentação Completa

Desenvolva um README completo para seu projeto, incluindo instruções de instalação, uso e exemplos práticos.

🔗 Recursos Adicionais

- [Documentação oficial do GitHub](#)
- [Conventional Commits](#)
- [Sphinx Napoleon para docstrings](#)