

# Ambiente de desenvolvimento

Disciplina: Programação aplicada à engenharia cartográfica

Maurício C. M. de Paulo - D.Sc.

25 de fevereiro de 2026

# Python (em 2026?)



# Bibliotecas úteis

Área	Bibliotecas Python
Matemática	NumPy, SciPy
Interface gráfica (GUI)	PyQt5, Tkinter, Kivys
Análises em SIG	PyQGIS, OGR, whitebox
Dados raster geográficos	GDAL, Fiona, Rasterio
Dataframes (tabelas)	Pandas, GeoPandas, Polars
Gráficos científicos	Matplotlib, Plotly
Backend Web	FastAPI, Flask, Django
Backend com interface	Taipy, Streamlit, Dash
Aprendizado de máquina	Scikit-learn, Keras, PyTorch, TensorFlow, JAX
Aprendizado profundo para geoinformação	torchgeo, rastervision, geoai-py

# Projetos

- Plugin Hello Map  
Carregar dados matriciais e vetoriais no QGIS (GDAL opcional)
- Abrir coordenadas da rede de monitoramento SIRGAS e converter para um arquivo vetorial  
Baixar e manipular dados tabulados com Pandas e escrever com Geopandas.
- Processing Plugin de extrair cotas de um Modelo Digital de Superfícies  
Manipular dados matriciais e vetoriais no QGIS
- Operações topológicas  
Processing Plugin com model builder para resolver um problema de geoprocessamento.
- Opcionais (não valem nota)
  - Classificação de imagens com RasterIO+Sklearn (XGBoost)
  - Interfaces web com Streamlit
  - Geoagentes de IA

# Estrutura dos projetos

- /docs

Diagrama de classes: Representa visualmente as principais classes que serão implementadas.

Fluxogramas: Representa uma visão geral de um procedimento importante.

- /src

Exemplos de código

- /tests

Testes unitários: Verificam partes do código utilizando entradas e saídas controladas.

# Avaliação

- 6 – Implementação

Entrega no github. Colocar o link para o repositório no Classroom.  
Em pares.

- 3 – Apresentação oral do código (no computador)

Alternado entre os dos dois membros do par.

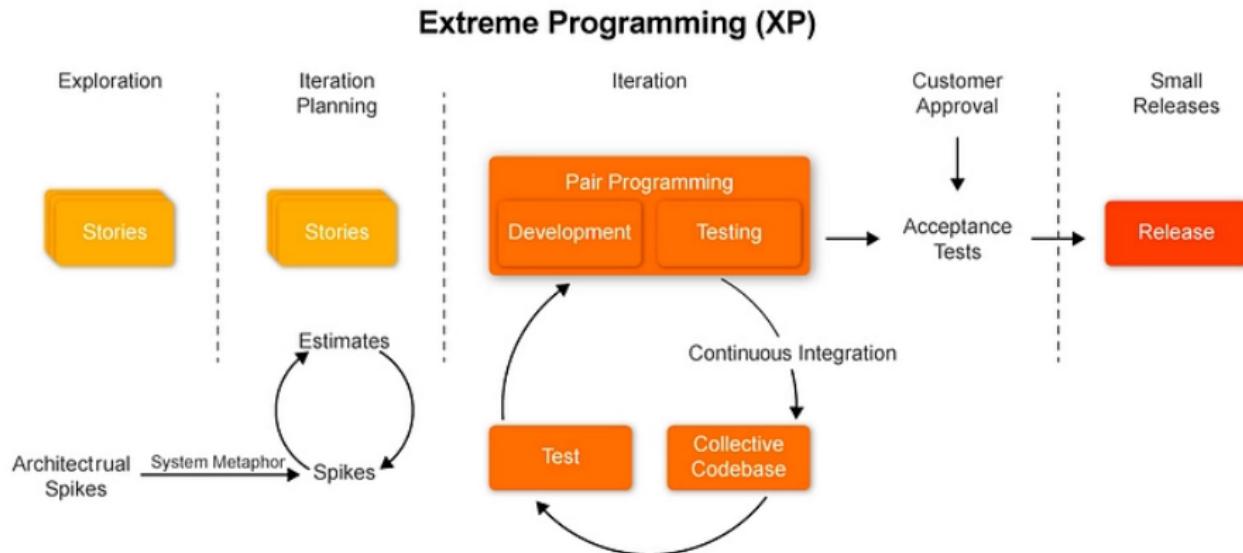
Os alunos explicam o que cada parte do código está fazendo e as decisões tomadas.

- 1 – Inovação

Ponto para ideias interessantes que o par tiver.

# Ciclos de desenvolvimento em pares

- Desenvolvimento ágil em ciclos de 2 semanas
- <https://ronjeffries.com/xprog/what-is-extreme-programming/>
- Programação em pares (Xtreme programming - XP)



# Instalação do Python (com ambientes virtuais)

Ambiente virtuais em Python:

- Isolamento de dependências entre projetos
- Evita conflitos de versões de bibliotecas
- Permite usar diferentes versões do Python no mesmo sistema
- Facilita a reprodução do ambiente em outros computadores
- Reduz riscos de quebrar aplicações existentes ao instalar novos pacotes

**Principal objetivo:** instalar GDAL e QGIS

# Instalação do Python (com ambientes virtuais)

## Gerenciadores de ambientes:

Ferramenta	Descrição	Ambiente	Observações
pip + venv	Pacotes Python	Virtual no projeto	Padrão do Python
uv	pip + venv rápido	Ambientes e projetos	Alto desempenho
Conda	Pacotes diversos	Estrutura Conda	Ciência de dados
Pixi	Alternativa ao Conda	Estilo Conda	Substituto moderno

- GDAL e QGIS: Pixi (recomendado) ou miniforge (conda).
- Deep learning: uv para pytorch, tensorflow, etc.

# Pixi: visão geral

**Pixi** é uma ferramenta moderna para:

- Gerenciar **ambientes** de desenvolvimento
- Gerenciar **dependências** de projetos
- Reproduzir projetos de forma consistente

**Principais características:**

- Baseado no ecossistema Conda
- Rápido e determinístico
- Um projeto = um ambiente

**Ideia central:** Cada projeto Python possui seu próprio ambiente isolado.

# Pixi: Criando um projeto Pixi

## Passo 1: criar o projeto

```
pixi init meu_projeto
```

Isso cria:

- Uma pasta do projeto
- O arquivo pixi.toml

## Arquivo central: pixi.toml

- Lista dependências
- Define versões
- Descreve o ambiente

# Pixi: Adicionando dependências

## Passo 2: adicionar pacotes ao projeto

```
pixi add python numpy pandas qgis
```

O Pixi:

- Resolve dependências automaticamente
- Cria o ambiente isolado
- Atualiza o pixi.toml

**Importante:**

- Não use pip install diretamente
- As dependências ficam documentadas no projeto

# Pixi: Executando código no ambiente

## Passo 3: usar o ambiente Pixi

```
pixi shell
```

Entra em um shell com o ambiente ativo.

Ou execute comandos diretamente:

```
pixi run python main.py  
pixi run qgis
```

## Benefícios:

- Mesmo ambiente para todos os usuários
- Menos erros de “funciona na minha máquina”
- Projeto mais organizado e reproduzível

# Inicialização de ambiente com UV

**O que é o UV?** Gerenciador moderno de:

- ambientes virtuais
- dependências Python

**Criando um novo projeto:**

```
uv init projeto_geo  
cd projeto_geo
```

**O que o UV cria automaticamente:**

- Arquivo pyproject.toml
- Ambiente virtual isolado
- Estrutura básica do projeto

**Ativação do ambiente:** O UV ativa o ambiente automaticamente ao executar comandos.

# Instalação do GeoPandas com UV

## Instalando o GeoPandas:

```
uv add geopandas
```

## O que acontece nesse passo:

- GeoPandas é adicionado ao `pyproject.toml`
- Dependências são resolvidas automaticamente
- O ambiente fica reproduzível

## Testando a instalação:

```
import geopandas as gpd  
print(gpd.__version__)
```

**Boa prática:** Sempre versionar o arquivo `pyproject.toml` no Git.

# Interface de desenvolvimento

## Integrated Development Environment (IDE) VSCode:

<https://vscode.com/>

Versão **código aberto** do VS Code.

Usaremos para:

- Escrever código Python
- Depuração (Debug)
- Controle de versões de código (Git)
- Cadernos Jupyter
- Desenhar diagramas
  - Fluxogramas
  - UML: Classes

Quais são as melhores IDE para Python? Forneça também os links para o site do projeto.



Existem várias IDEs para Python, cada uma com suas próprias características e vantagens. Aqui estão algumas das melhores opções:



### IDEs Comerciais e Gratuitas

- PyCharm: Uma IDE completa com recursos como destaque de sintaxe, preenchimento automático, depuração e gerenciamento de projetos. Disponível em versões comunitária, educacional e profissional.<sup>1</sup>
  - Site: (link unavailable)
- Visual Studio Code (VS Code): Um editor de código leve e extensível, com suporte a diversas linguagens, incluindo Python. Oferece recursos como depuração, gerenciamento de código e extensões.<sup>1</sup>
  - Site: (link unavailable)
- Spyder: Uma IDE gratuita e de código aberto, com recursos como destaque de sintaxe, preenchimento automático, depuração e visualização de dados.
  - Site: (link unavailable)



### IDEs para Desenvolvimento Web e Ciência de Dados

- Jupyter Notebook: Uma ferramenta de código aberto para ciência de dados e aprendizado de máquina, com suporte a notebooks interativos.<sup>1</sup>
  - Site: (link unavailable)
- Google Colab: Um serviço de nuvem gratuito para desenvolvimento de machine learning e inteligência artificial, com suporte a notebooks interativos.<sup>1</sup>
  - Site: (link unavailable)



### Outras Opções

- IDLE: Uma IDE básica e multiplataforma, com recursos como destaque de sintaxe e depuração.<sup>1</sup>
  - Site: (link unavailable)

# Interface de desenvolvimento

Caso ainda não tenha instalado, essa é a hora de instalar o VSCodium.

Acessar <https://github.com/VSCodium/vscodium/releases>

- Windows: Baixar o arquivo VSCodiumUserSetup-x64-1.108.20787.exe e executar.
- Linux: Baixar o arquivo .deb e instalar com:

```
dpkg -i codium_1.108.20787_amd64.deb
```

# Copilotos de desenvolvimento

Origem: StackOverflow <https://stackoverflow.com/>

- ChatGPT <https://chatgpt.com/>
- Github Copilot <https://github.com/features/copilot>
- Gemini <https://gemini.google.com>
- Llama: <https://www.meta.ai/> (pesos e código abertos)
- **Instalação local:** Ollama <https://ollama.com/>

Entre outros, alguns especialistas.

Não confie cegamente no trabalho de outra pessoa.

**Muito menos de uma IA.**

# VSCodium: Copilotos de desenvolvimento

Extensões disponíveis no VSCode/VSCodium:

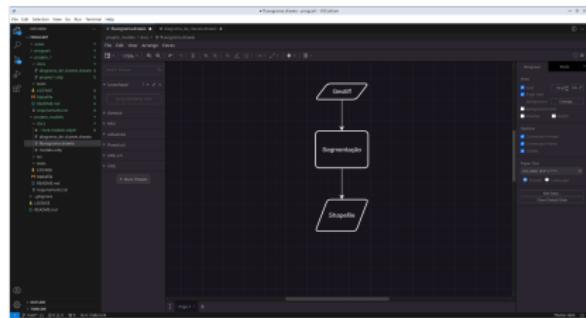
- Github Copilot (nativo)
- Gemini
- Codegpt
- Continue.dev + Ollama **Instalação local**

Cuidados:

- Seu código é compartilhado com o dono do serviço (exceto se o serviço for seu).
- É fácil consumir todos os limites de uso das versões gratuitas.

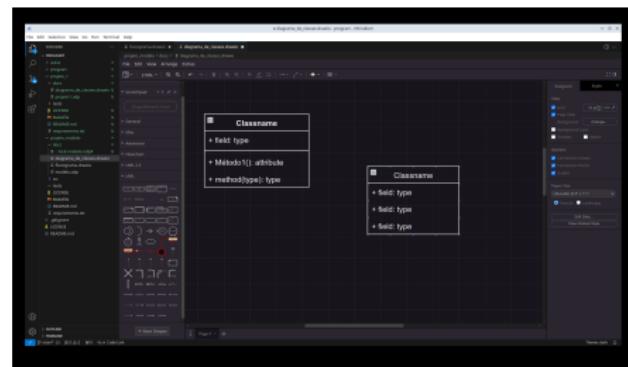
# VSCodium: Fluxograma

- Pasta: projeto/docs/  
fluxograma.drawio
- Abrir com:  
<https://www.draw.io.com/>
- VSCodium Draw.io Integration  
(Author: Hediet)
- Tutorial de edição: <https://www.draw.io.com/doc/getting-started-basic-flow-chart>



# VSCodium: Diagramas de classes (UML)

- Pasta:  
`projeto/docs/diagrama_de_classes.drawio`
- Abrir com:  
<https://www.draw.io/>
- VSCodium Draw.io Integration  
(Author: Hediet)
- Tutorial de edição:  
<https://www.draw.io/blog/uml-class-diagramss>



# VSCodium: Depuração (Debug)

O **debug** permite executar o programa passo a passo, inspecionando valores e o fluxo do código.

## Passos básicos:

- Abrir o arquivo Python no VSCodium
- Definir **breakpoints** (clique à esquerda da linha)
- Iniciar o modo Debug

## Durante o debug, é possível:

- Executar linha a linha (*Step Over / Step Into*)
- Ver valores das variáveis em tempo real
- Avaliar expressões
- Identificar erros de lógica

## Alterar as configurações do VSCodium devido a um bug do Pixi

- File - Preferences - Settings
- Python - Terminal: Activate Environment - Disable



# VSCodium: Debug com Pixi

**Objetivo:** Depurar um código Python usando o **ambiente Pixi** dentro do VSCodium.

## Passo 1 — Abrir o projeto Pixi

- Abra a pasta do projeto no VSCodium
- O arquivo `pixi.toml` deve estar na raiz

## Passo 2 — Ativar o ambiente Pixi no terminal

```
pixi shell
```

Isso garante que o Python e as dependências corretas estejam ativas.

## Passo 3 — Configurar o Debug

No menu *Run and Debug*, escolha:

- **Python File**
- O VSCodium usará o Python do ambiente Pixi

## Passo 4 — Depurar

- Coloque breakpoints no código
- Inicie o Debug