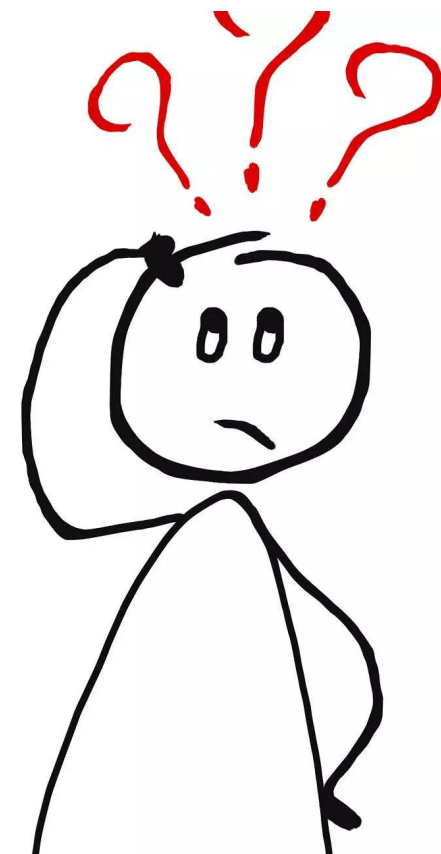


PROGRAMAÇÃO WEB II

Curso Técnico Integrado em Informática
Lucas Sampaio Leite



O que são pacotes em Python?

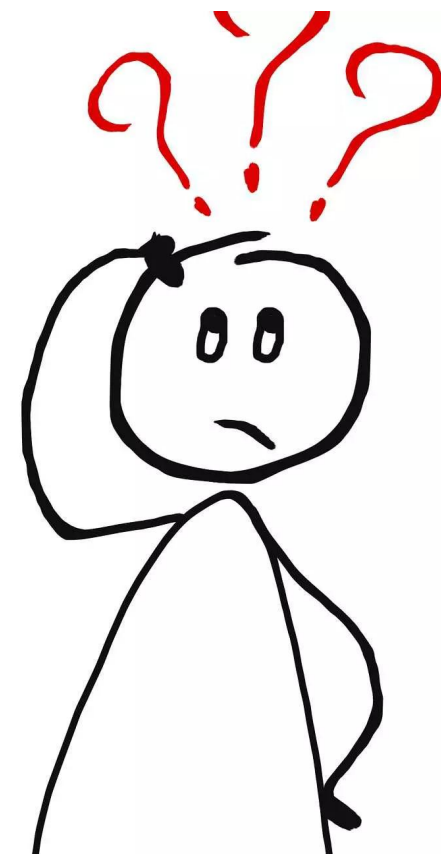


O que são pacotes em Python?

- Em Python, pacotes (*packages*) são uma forma de organizar módulos em estruturas hierárquicas utilizando diretórios.
 - Módulo: é um arquivo .py contendo definições de funções, classes, variáveis, estruturas de controle, etc.
- Pacotes servem para organizar o código de forma modular, facilitando a reutilização, manutenção e legibilidade.

Pacote é um diretório que contém um conjunto de módulos e, opcionalmente, subpacotes.

O que é o pip do Python?



O que é o pip do Python?

- pip ("Pip Installs Packages") é o gerenciador de pacotes oficial do Python. Ele permite a instalação, atualização e remoção de pacotes de terceiros disponíveis no Python Package Index (PyPI).
- O Python Package Index (PyPI) é um repositório de software para a linguagem de programação Python.



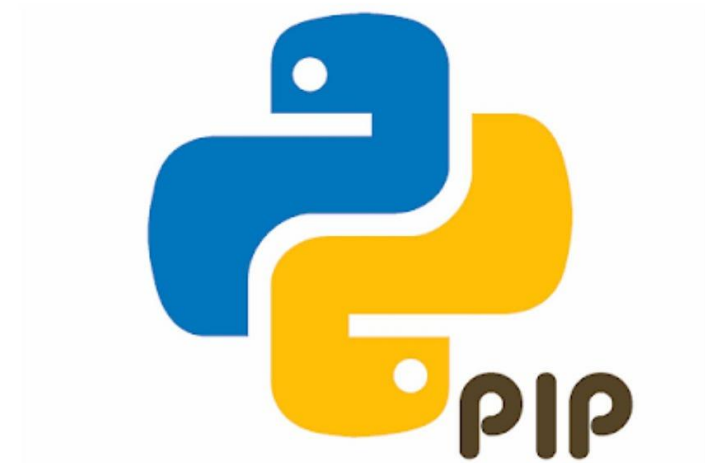
O Python Package Index (PyPI) é um repositório de software para a linguagem de programação Python.

O PyPI ajuda você a encontrar e instalar softwares desenvolvidos e compartilhados pela comunidade do Python. [Saiba mais sobre a instalação de pacotes](#).

Os autores de pacotes usam o PyPI para distribuir seu software. [Saiba como empacotar seu código Python para PyPI](#).

O que é o pip do Python?

- Com o pip, pode-se adicionar bibliotecas de terceiros ao projeto, como:
 - numpy para cálculo numérico
 - pandas para análise de dados
 - requests para fazer requisições HTTP
 - flask ou Django para criar APIs web



Porque o pip é tão poderoso?

- Reaproveitamento de código
 - Não reinventar a roda! Facilita a reutilização de código já testado e aprovado por outros desenvolvedores.
- Facilita o desenvolvimento
 - Instalação de bibliotecas com um comando e rápida utilização.
- Gerenciamento de dependências
 - O pip se encarrega de baixar os pacotes corretos, instalar versões compatíveis, atualizar/desinstalar quando necessário.
- Integração com ambientes virtuais
 - Usar pip com ambientes virtuais (venv, pipenv, poetry, etc.) garante que as dependências de um projeto não interfiram em outros projetos.
- Portabilidade
 - Uso de requirements.txt com todas as bibliotecas do seu projeto.

Comando para uso do pip

- Instalar um pacote: `pip install nome_do_pacote`
- Instalar uma versão específica do pacote: `pip install nome_do_pacote==nome_da-versao`
- Atualizar um pacote: `pip install --upgrade nome_do_pacote`
- Desinstalar um pacote: `pip uninstall nome_pacote`
- Ver os pacotes instalados: `pip list`
- Ver informações sobre um pacote: `pip show nome_pacote`

Exemplos de uso do pip

- Suponha que você está desenvolvendo uma aplicação de plotagem de gráficos. O objetivo é que, dada uma lista de dados numéricos, o sistema seja capaz de gerar automaticamente um gráfico de barras para representar visualmente esses valores.
- A biblioteca matplotlib é uma das bibliotecas mais poderosas e populares para visualização de dados em Python. Ela permite criar gráficos e visualizações de uma maneira simples e eficiente.



Exemplos de uso do pip

- <https://pypi.org/>



Ajuda Patrocinadores Entrar Registrar

Encontre, instale e publique pacotes Python com o Python Package Index

Ou [navegue por projetos](#)

628.982 projetos 6.844.130 lançamentos 14.001.103 arquivos 927.796 usuários



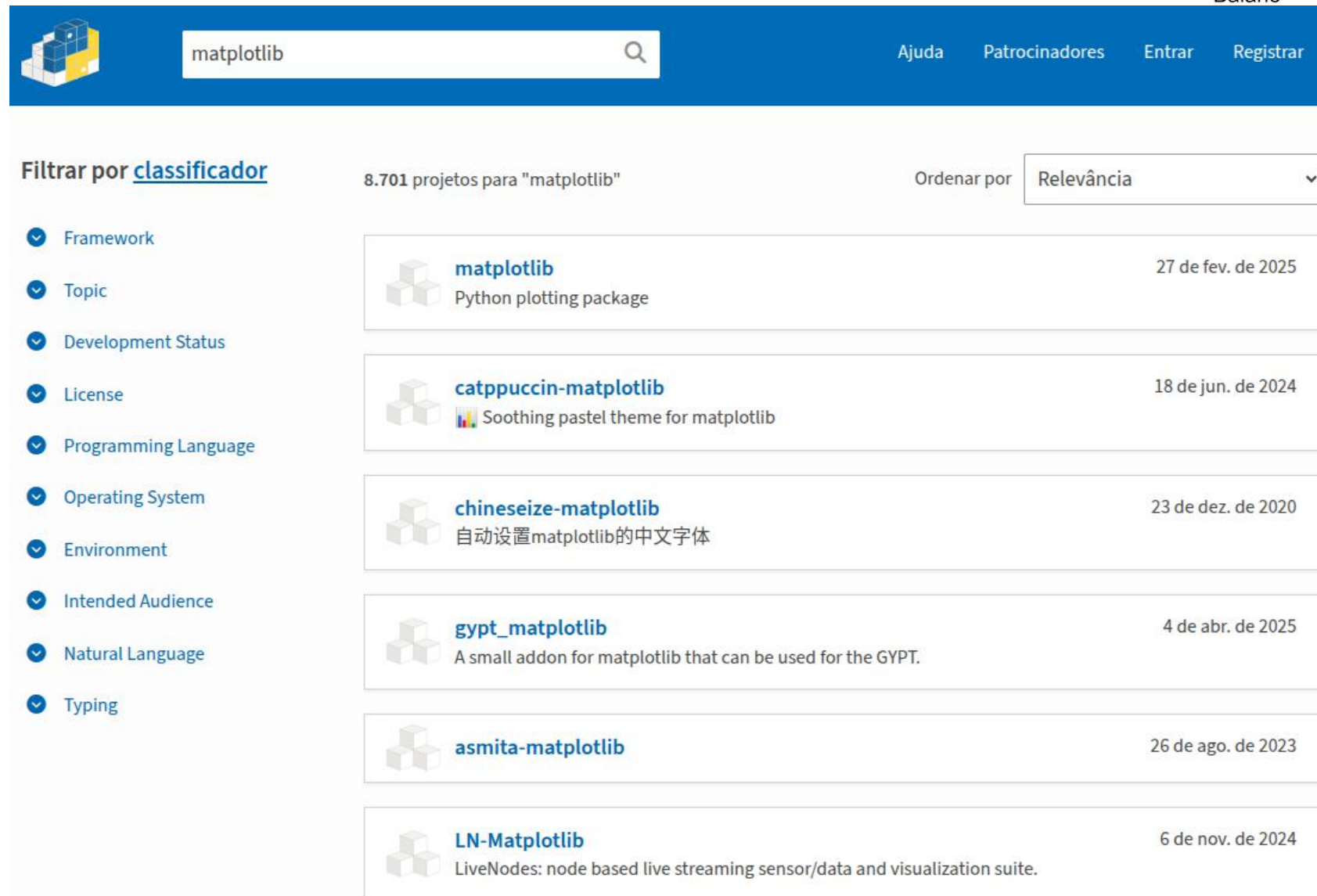
O Python Package Index (PyPI) é um repositório de software para a linguagem de programação Python.

O PyPI ajuda você a encontrar e instalar softwares desenvolvidos e compartilhados pela comunidade do Python. [Saiba mais sobre a instalação de pacotes](#).

Os autores de pacotes usam o PyPI para distribuir seu software. [Saiba como empacotar seu código Python para PyPI](#).

Exemplos de uso do pip

- <https://pypi.org/>



The screenshot shows the PyPI search results for the package 'matplotlib'. The search bar at the top contains the text 'matplotlib'. Below the search bar, there are filters on the left, a summary of results, and a list of packages.

Filtrar por classificador

- Framework
- Topic
- Development Status
- License
- Programming Language
- Operating System
- Environment
- Intended Audience
- Natural Language
- Typing

8.701 projetos para "matplotlib"

Ordenar por Relevância

Package Name	Description	Last Update
matplotlib	Python plotting package	27 de fev. de 2025
catpuccin-matplotlib	Soothing pastel theme for matplotlib	18 de jun. de 2024
chineseize-matplotlib	自动设置matplotlib的中文字体	23 de dez. de 2020
gypt_matplotlib	A small addon for matplotlib that can be used for the GYPT.	4 de abr. de 2025
asmita-matplotlib		26 de ago. de 2023
LN-Matplotlib	LiveNodes: node based live streaming sensor/data and visualization suite.	6 de nov. de 2024

Exemplos de uso do pip

- <https://pypi.org/>



The screenshot shows the PyPI page for matplotlib 3.10.1. The header is blue with a search bar and navigation links. The main section features the package name 'matplotlib 3.10.1' and a 'pip install matplotlib' button. Below this, it identifies the package as a 'Python plotting package'. The left sidebar contains navigation links like 'Descrição do projeto', 'Histórico de lançamentos', and 'Baixar arquivos'. The right section, 'Descrição do projeto', includes a row of badges for pypi, conda-forge, downloads, and powered by NumFOCUS, followed by links for help, forum, chat, issue tracking, github, and PR. Below these are status badges for tests, azure pipelines, build, codecov, and version scheme. The large 'matplotlib' logo is prominently displayed, with a description stating it's a comprehensive library for creating static, animated, and interactive visualizations in Python. At the bottom, there are four small images showing different types of plots: a bell curve, a heatmap, a line plot, and a 3D surface plot.

Pesquisar projetos

Ajuda Patrocinadores Entrar Registrar

matplotlib 3.10.1

✓ Versão mais recente

Lançamento em: 27 de fev. de 2025

`pip install matplotlib`

Python plotting package

Navegação

- Descrição do projeto
- Histórico de lançamentos
- Baixar arquivos

Detalhes verificaddos

Estes detalhes foram [verificados pela PyPI](#)

Links do projeto

- Bug Tracker
- Source Code

Owner

- Matplotlib

Estatísticas do GitHub

- Repositório
- Estrelas: 21085
- Forks: 7838

Descrição do projeto

pypi v3.10.1 conda-forge v3.10.1 downloads 84M/month powered by NumFOCUS

help forum discourse chat on glitter issue tracking github PR Welcome

Tests failing Azure Pipelines succeeded build unknown codecov 88% version scheme EffVer

matplotlib

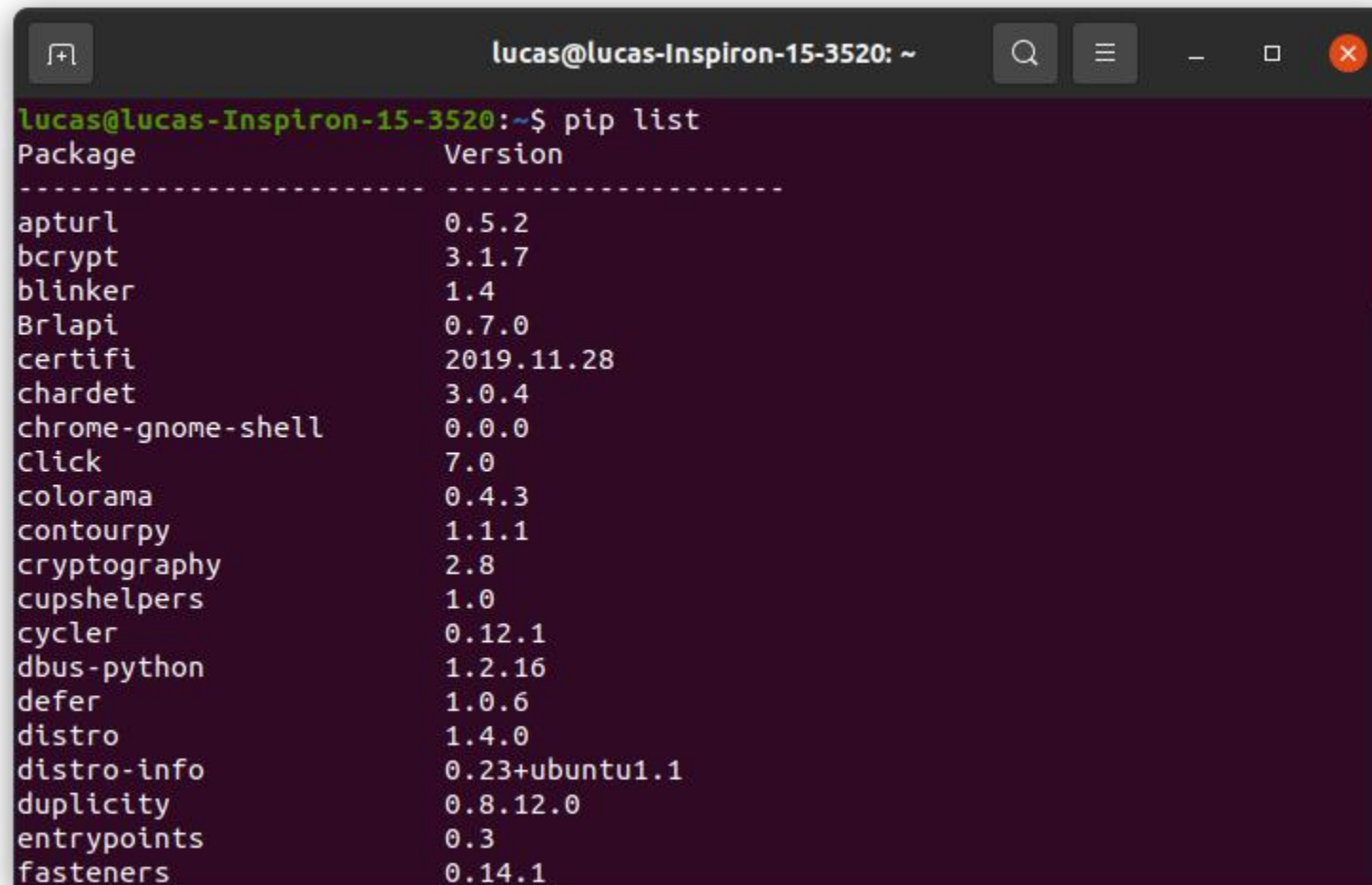
Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Check out our [home page](#) for more information.



Exemplos de uso do pip

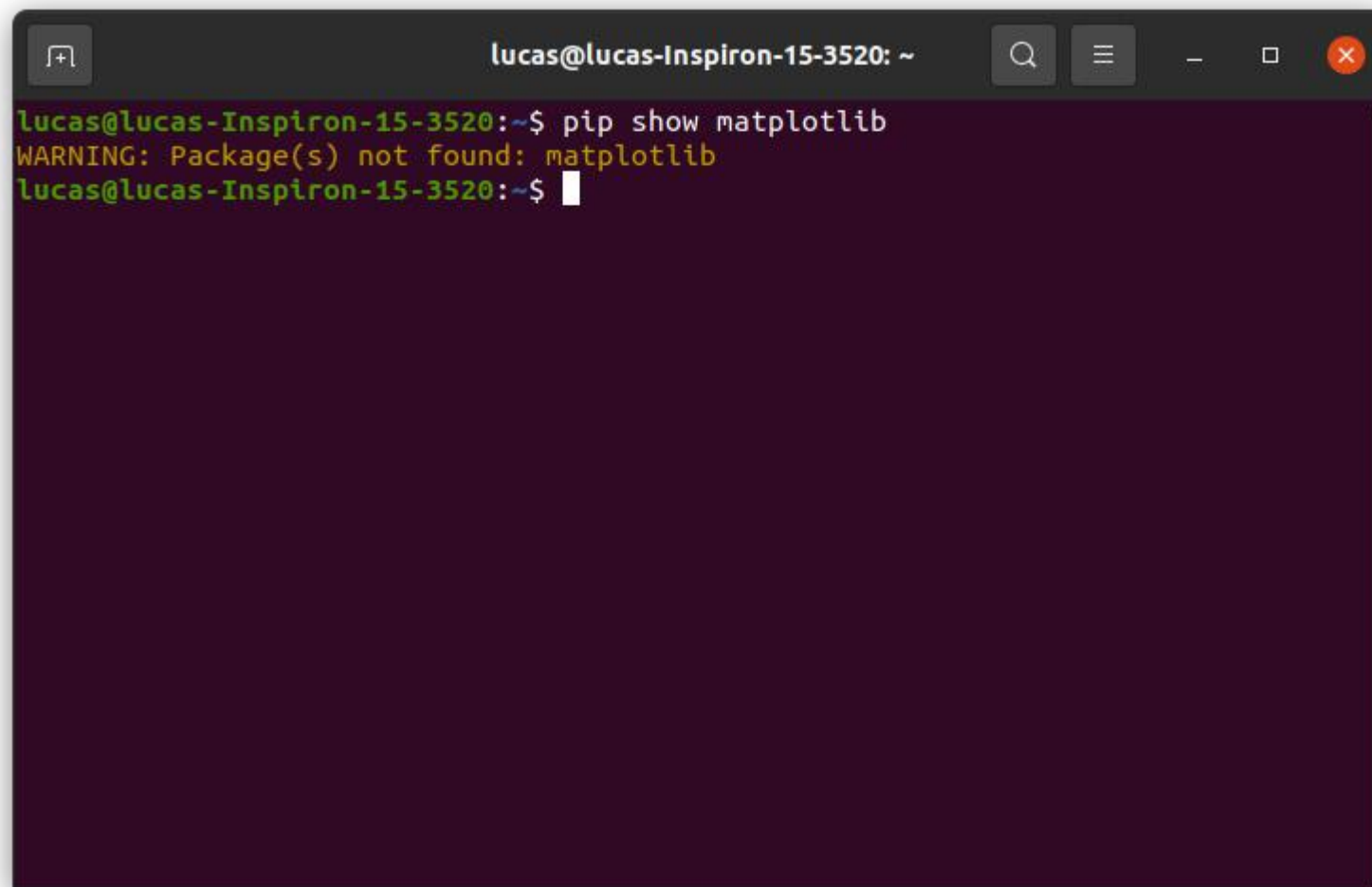
- Verificando as bibliotecas existentes: **pip list**



```
lucas@lucas-Inspiron-15-3520: ~$ pip list
Package            Version
-----
apturl              0.5.2
bcrypt              3.1.7
blinker             1.4
Brlapi              0.7.0
certifi             2019.11.28
chardet             3.0.4
chrome-gnome-shell  0.0.0
Click               7.0
colorama            0.4.3
contourpy           1.1.1
cryptography        2.8
cupshelpers         1.0
cyclcr              0.12.1
dbus-python         1.2.16
defer               1.0.6
distro              1.4.0
distro-info         0.23+ubuntu1.1
duplicity           0.8.12.0
entrypoints         0.3
fasteners           0.14.1
```

Exemplos de uso do pip

- Exibindo informações detalhadas sobre um pacote Python que já está instalado no ambiente: **pip show**



```
lucas@lucas-Inspiron-15-3520: ~  
lucas@lucas-Inspiron-15-3520:~$ pip show matplotlib  
WARNING: Package(s) not found: matplotlib  
lucas@lucas-Inspiron-15-3520:~$
```


Exemplos de uso do pip

- Instalando o pacote:

```
lucas@lucas-Inspiron-15-3520: ~  
lucas@lucas-Inspiron-15-3520:~$ pip install matplotlib  
Collecting matplotlib  
  Using cached matplotlib-3.7.5-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (9.2 MB)  
Collecting numpy<2,>=1.20  
  Using cached numpy-1.24.4-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)  
Requirement already satisfied: pillow>=6.2.0 in /usr/lib/python3/dist-packages (from matplotlib) (7.0.0)  
Requirement already satisfied: pyparsing>=2.3.1 in ~/.local/lib/python3.8/site-packages (from matplotlib) (3.1.2)  
Requirement already satisfied: contourpy>=1.0.1 in ~/.local/lib/python3.8/site-packages (from matplotlib) (1.1.1)  
Requirement already satisfied: kiwisolver>=1.0.1 in ~/.local/lib/python3.8/site-packages (from matplotlib) (1.4.5)  
Requirement already satisfied: python-dateutil>=2.7 in /usr/lib/python3/dist-packages (from matplotlib) (2.7.3)  
Requirement already satisfied: importlib-resources>=3.2.0; python_version < "3.10" in ~/.local/lib/python3.8/site-packages (from matplotlib) (6.4.0)  
Requirement already satisfied: cycler>=0.10 in ~/.local/lib/python3.8/site-packages (from matplotlib) (0.12.1)  
Requirement already satisfied: fonttools>=4.22.0 in ~/.local/lib/python3.8/site-packages (from matplotlib) (4.52.1)  
Requirement already satisfied: packaging>=20.0 in ~/.local/lib/python3.8/site-packages (from matplotlib) (24.0)  
Requirement already satisfied: zipp>=3.1.0; python_version < "3.10" in ~/.local/lib/python3.8/site-packages (from importlib-resources>=3.2.0; python_version < "3.10"->matplotlib) (3.18.2)  
Installing collected packages: numpy, matplotlib  
Successfully installed matplotlib-3.7.5 numpy-1.24.4  
lucas@lucas-Inspiron-15-3520:~$
```

Exemplos de uso do pip

- Exibindo informações detalhadas sobre um pacote Python que já está instalado no ambiente: **pip show**

```
lucas@lucas-Inspiron-15-3520: ~  
lucas@lucas-Inspiron-15-3520:~$ pip show matplotlib  
Name: matplotlib  
Version: 3.7.5  
Summary: Python plotting package  
Home-page: https://matplotlib.org  
Author: John D. Hunter, Michael Droettboom  
Author-email: matplotlib-users@python.org  
License: PSF  
Location: /home/lucas/.local/lib/python3.8/site-packages  
Requires: kiwisolver, pyparsing, contourpy, numpy, pillow, importlib-resources,  
fonttools, python-dateutil, cycler, packaging  
Required-by:  
lucas@lucas-Inspiron-15-3520:~$
```


Exemplos de uso do pip

- Importando e usando a biblioteca:

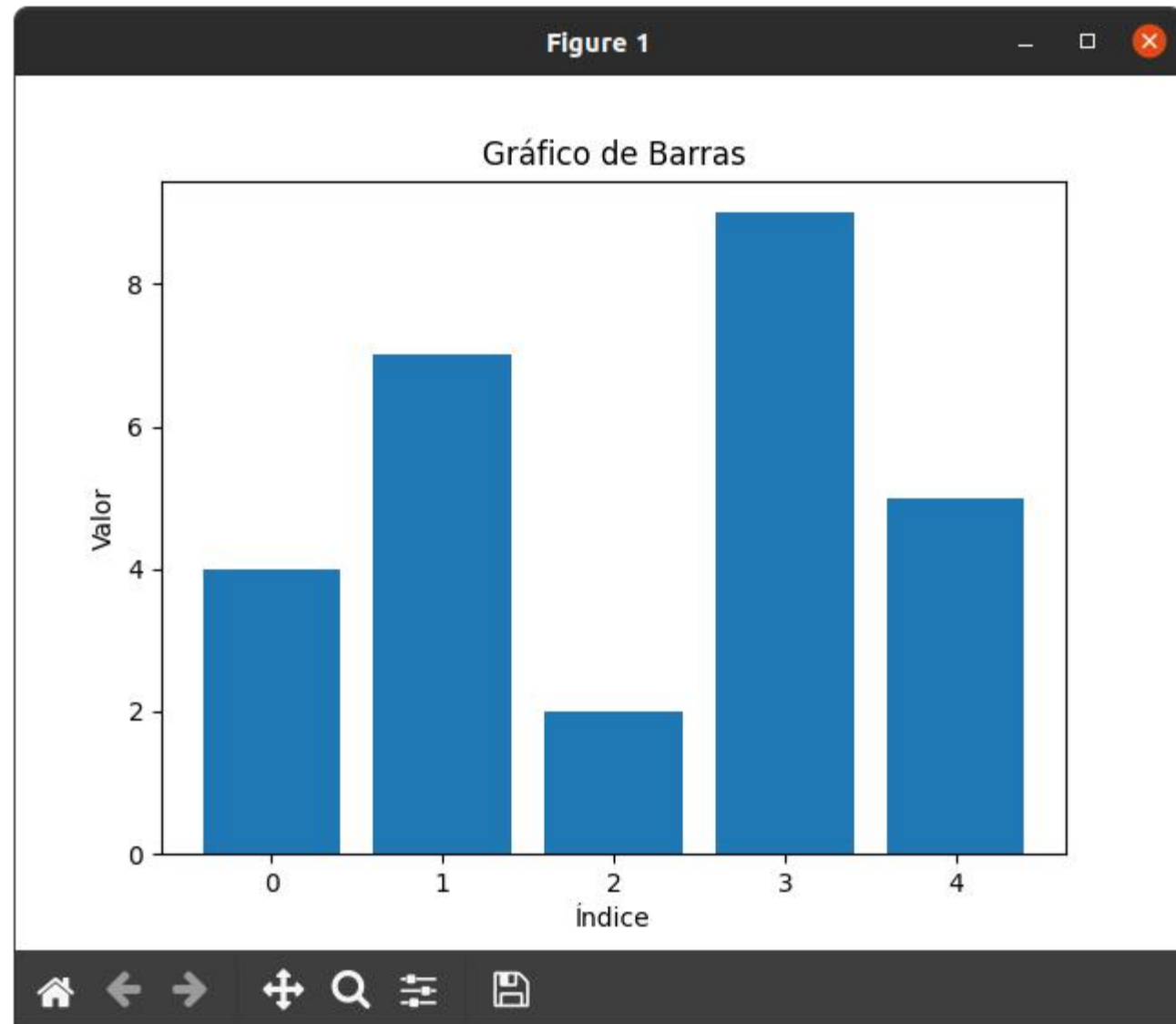
```
import matplotlib.pyplot as plt

valores = [4, 7, 2, 9, 5]

plt.bar(range(len(valores)), valores)

plt.title('Gráfico de Barras')
plt.xlabel('Índice')
plt.ylabel('Valor')

plt.show()
```



Problema de Usar o pip instalando globalmente

Computador

Python 3.10

Problema de Usar o pip instalando globalmente

Computador

Python 3.10

Projeto 1
requer
Django 5.2

Problema de Usar o pip instalando globalmente

Computador

Python 3.10
Django 5.2

Projeto 1
requer
Django 5.2

Problema de Usar o pip instalando globalmente

Computador

Python 3.10
Django 5.2

Projeto 1
requer
Django 5.2

Projeto 2
requer
NumPy 2.2

Problema de Usar o pip instalando globalmente

Computador

Python 3.10
Django 5.2
NumPy 2.2

Projeto 1
requer
Django 5.2

Projeto 2
requer
NumPy 2.2

Problema de Usar o pip instalando globalmente

Computador

Python 3.10
Django 5.2
NumPy 2.2

Projeto 1
requer
Django 5.2

Projeto 2
requer
NumPy 2.2

Projeto 3
requer
Flask 3.1

Problema de Usar o pip instalando globalmente

Computador

Python 3.10
Django 5.2
NumPy 2.2
Flask 3.1

Projeto 1
requer
Django 5.2

Projeto 2
requer
NumPy 2.2

Projeto 3
requer
Flask 3.1

Problema de Usar o pip instalando globalmente

Computador

Python 3.10
Django 5.2
NumPy 2.2
Flask 3.1

Projeto 1
requer
Django 5.2

Projeto 2
requer
NumPy 2.2

Projeto 3
requer
Flask 3.1

Projetos grandes
normalmente exigem
muitas dependências!!!

Como ficaria essa
situação?

Problema de Usar o pip instalando globalmente

Computador

Python 3.10
Django 5.2
NumPy 2.2
Flask 3.1

Projeto 1
requer
Django 5.2

Projeto 2
requer
NumPy 2.2

Projeto 3
requer
Flask 3.1

Digamos que o desenvolvedor precise trabalhar em um Projeto 4 que requer uma versão anterior ao Django 2.0. Como ficaria a situação?

Problema de Usar o pip instalando globalmente

- Quando se instala pacotes Python globalmente (usando o comando **pip install** sem o uso de ambientes virtuais), pode-se enfrentar vários problemas, especialmente ao trabalhar em diferentes projetos ou com diferentes versões de pacotes.
- Conflito de Dependências:
 - Se diferentes projetos exigem versões diferentes de uma mesma biblioteca, o pip global pode instalar apenas uma versão por vez, o que pode causar conflitos. Por exemplo, se o projeto A precisa da versão 1.0 de uma biblioteca e o projeto B precisa da versão 2.0, não será possível ter as duas versões instaladas globalmente.

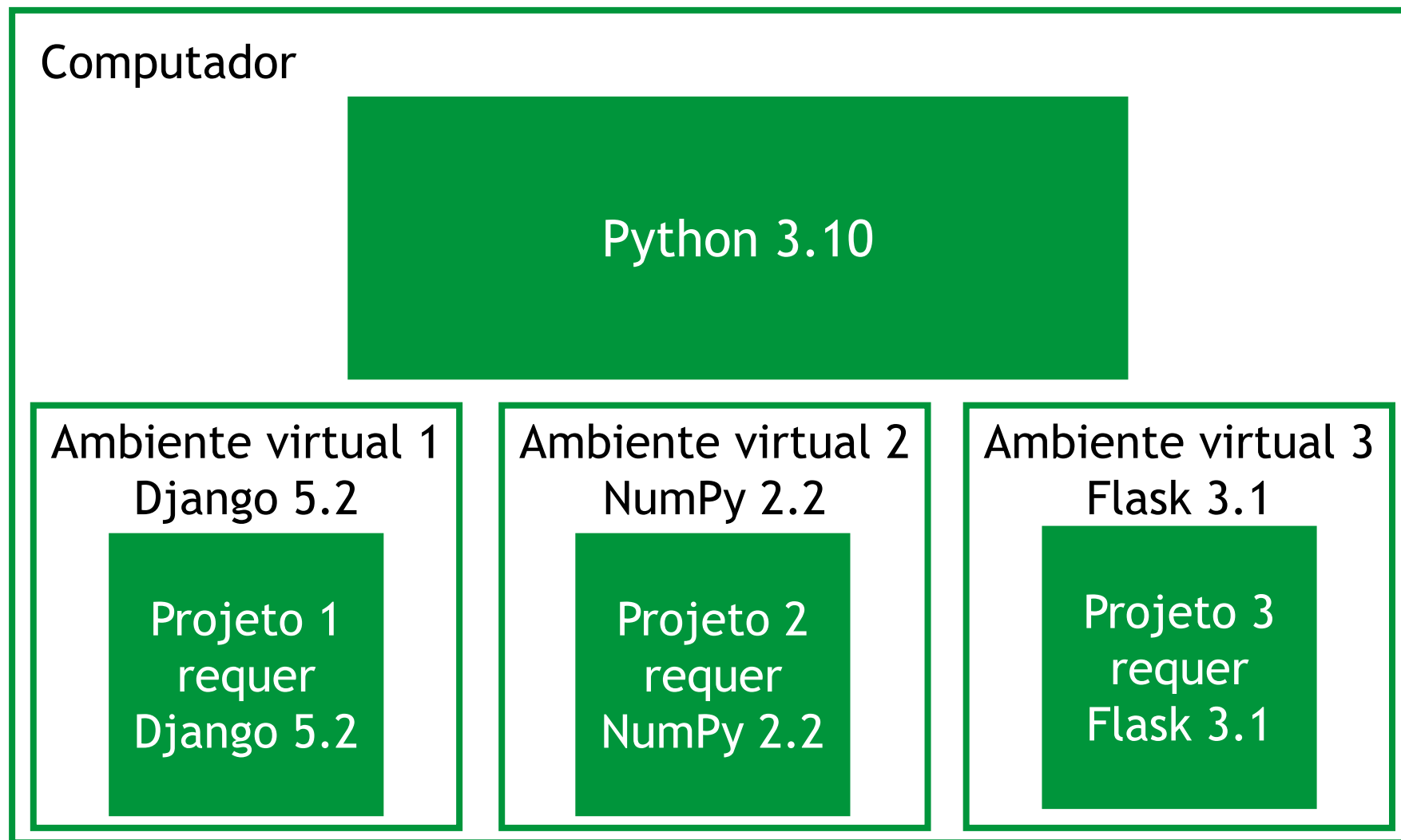
Problema de Usar o pip instalando globalmente

- Permissões de Sistema:
 - Quando se instala pacotes globalmente, dependendo do sistema operacional, pode ser necessário ter permissões de administrador.
- Impacto em Outros Projetos:
 - Instalar pacotes globalmente pode afetar outros projetos ou usuários no sistema. Atualizações ou mudanças em uma biblioteca global podem quebrar outros projetos que dependem de versões específicas de pacotes.
- Dificuldade na Reprodutibilidade:
 - Ao não isolar as dependências de um projeto, torna-se difícil garantir que outro desenvolvedor, ou até mesmo você em outra máquina, consiga reproduzir o ambiente de desenvolvimento com as mesmas versões de pacotes, resultando em "works on my machine" (funciona na minha máquina, mas não na sua).

Ambientes virtuais

- Um ambiente virtual em Python é um espaço isolado criado para um projeto específico, onde você pode instalar pacotes e bibliotecas sem afetar o sistema global ou outros projetos.
- Ele funciona como uma "bolha" que contém seu próprio interpretador Python e suas próprias dependências.

Ambientes virtuais



Vantagens usar ambientes virtuais

- Isolamento de dependências: Evita conflitos entre projetos.
- Reprodutibilidade: Você pode compartilhar o projeto com outras pessoas sem preocupações (requirements.txt ou Pipfile.lock).
- Segurança: Nada é instalado no sistema global.
- Organização: Cada projeto fica com suas próprias ferramentas e bibliotecas.

venv

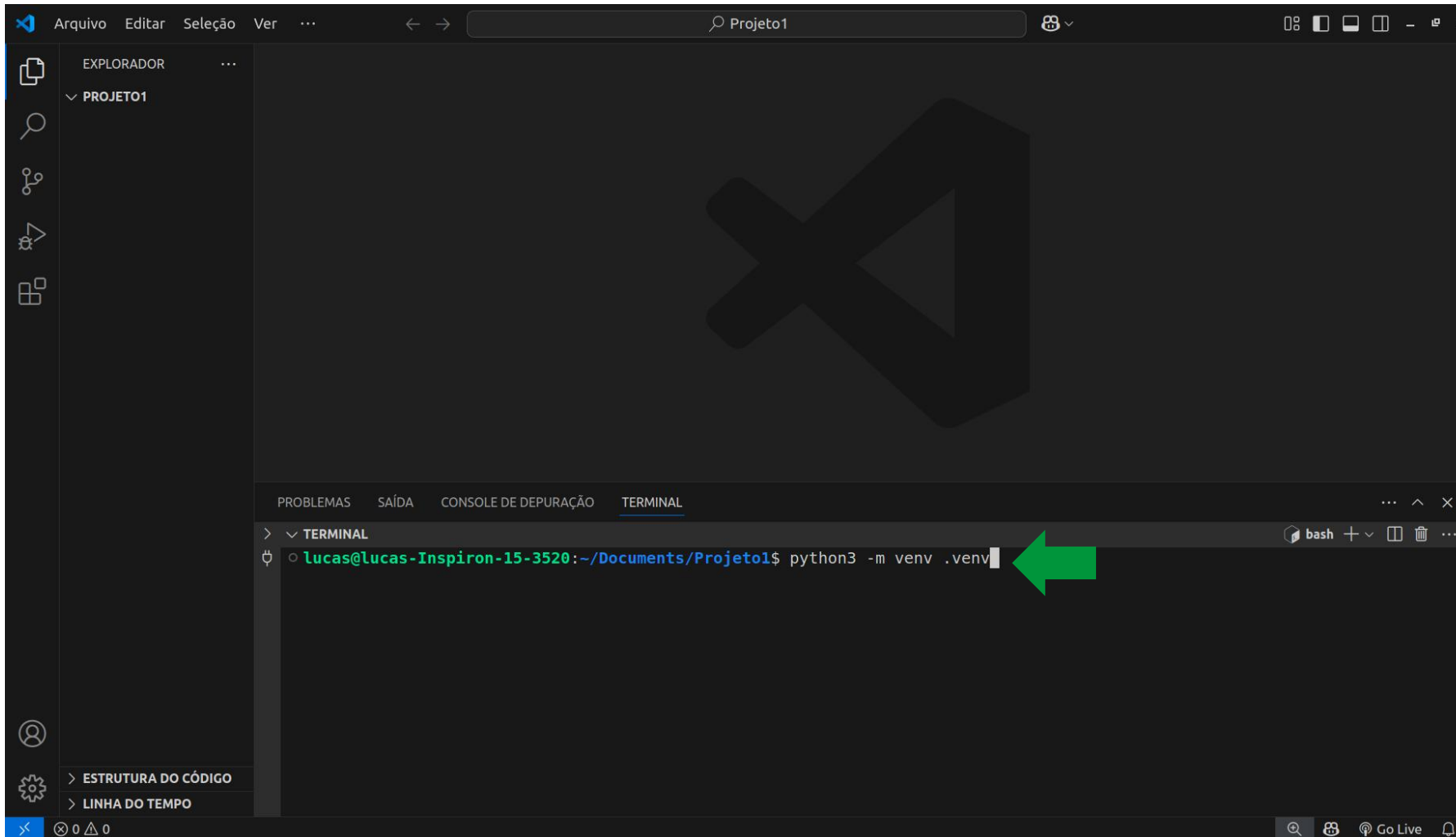
- venv é um módulo da biblioteca padrão do Python que permite criar ambientes virtuais de forma simples e rápida, sem precisar instalar nada extra.
- O venv cria uma cópia leve e isolada do interpretador Python e da estrutura de pacotes, permitindo que você instale bibliotecas específicas para um projeto sem interferir no sistema ou em outros projetos.

venv

- Criar o ambiente: `python -m venv nome_do_ambiente`
 - Exemplo: `python -m venv .venv`
- Ativar o ambiente:
 - Windows: `nome_do_ambiente\Scripts\activate`
 - Linux/macOS: `source nome_do_ambiente/bin/activate`
- Instalar pacotes localmente:
 - `pip install nome_do_pacote`
- Desativar o ambiente:
 - `deactivate`

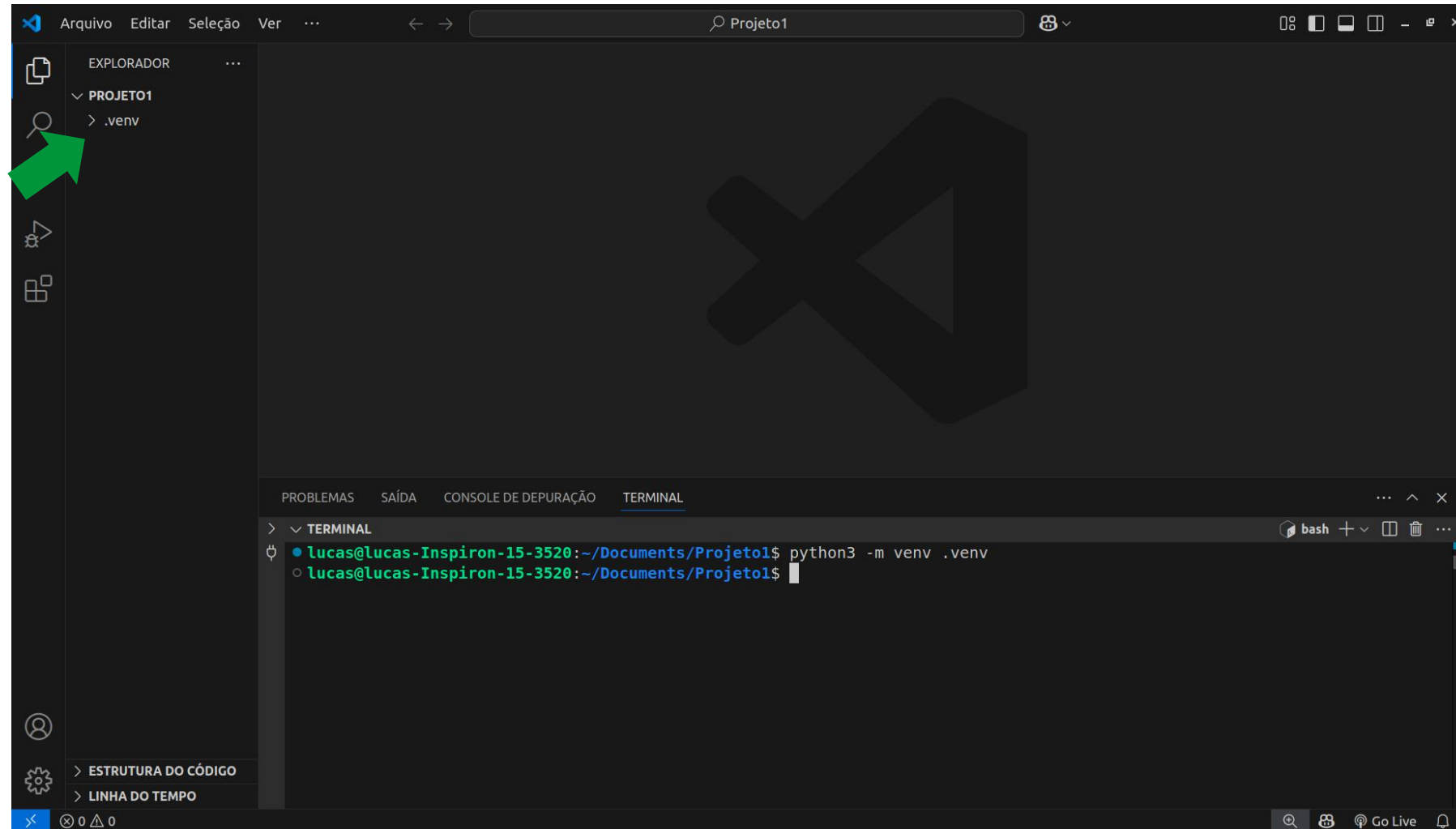
venv

- Criar o ambiente: `python -m venv .venv`



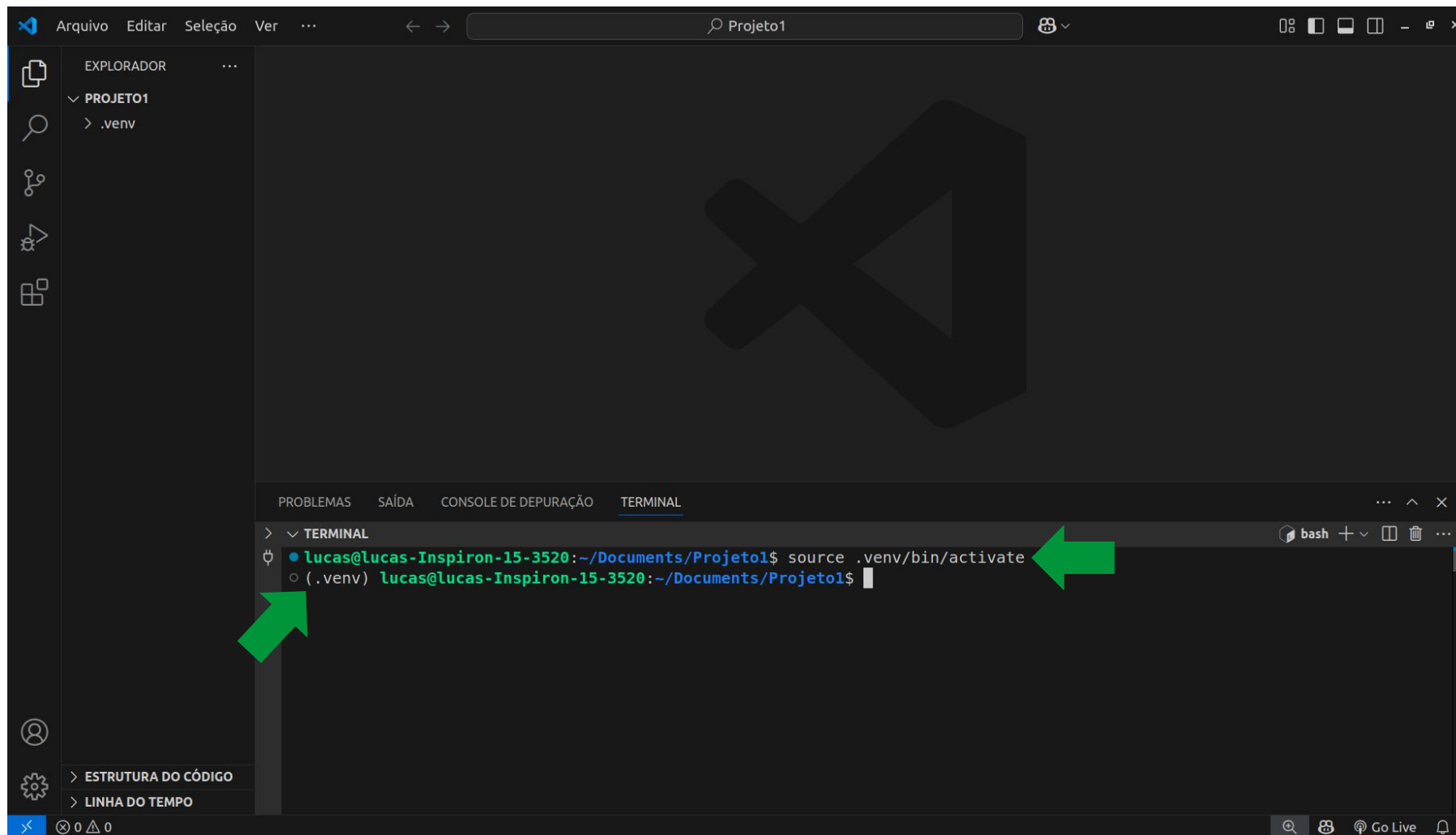
venv

- Criar o ambiente: `python -m venv .venv`



venv

- Ativar o ambiente: `.venv\Scripts\activate` ou `source .venv/bin/activate`

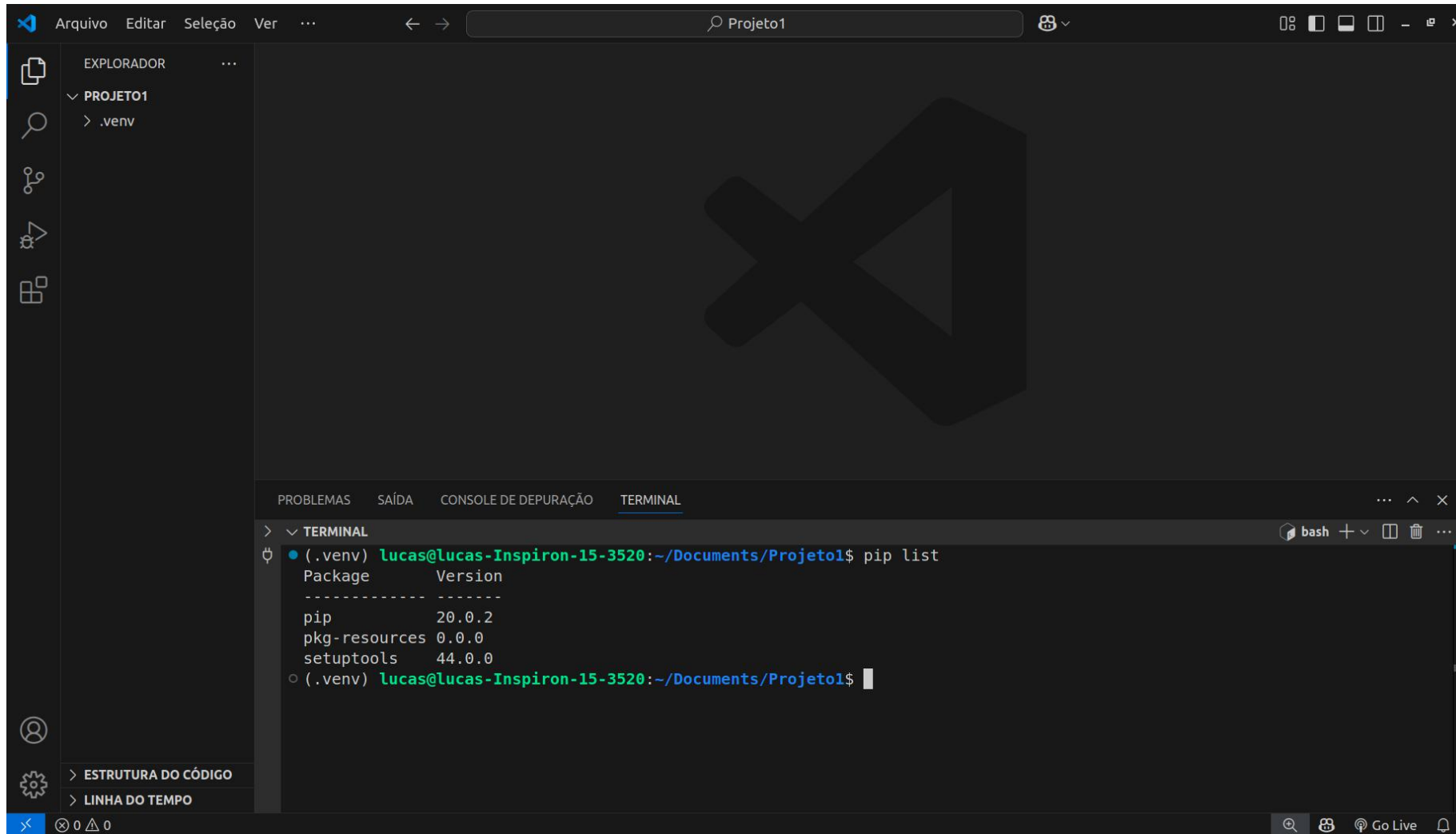


The screenshot shows the Visual Studio Code interface with a terminal window open at the bottom. The terminal shows the command `source .venv/bin/activate` being executed, and the prompt changes to `(.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$`. Two green arrows point to the command and the resulting prompt.

```
lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$ source .venv/bin/activate
(.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$
```

venv

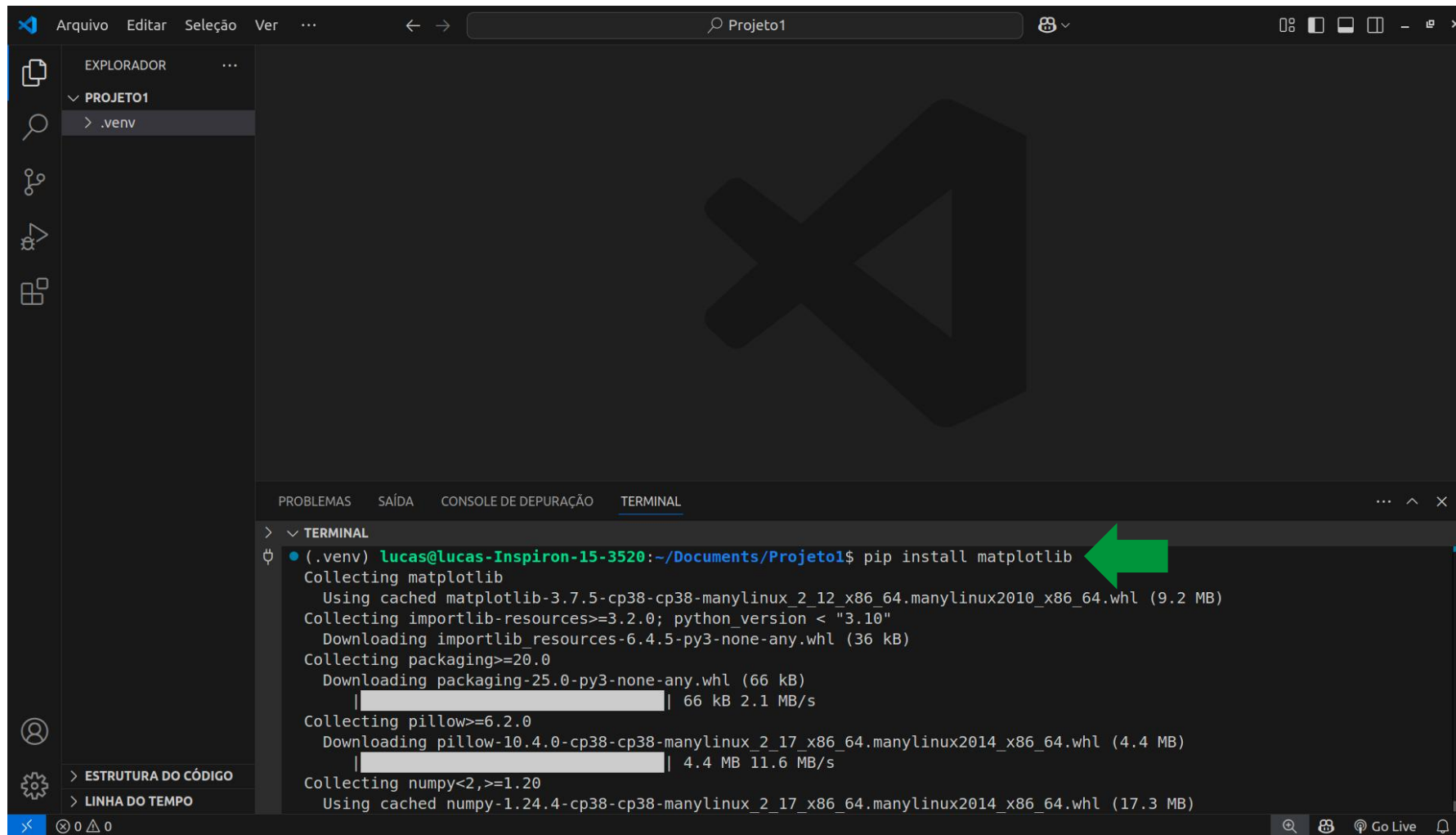
- Listando as dependências do ambiente virtual: **pip list**



The screenshot shows the Visual Studio Code interface with a terminal window open at the bottom. The terminal is running the command `pip list` in a virtual environment. The output shows the installed packages and their versions:

```
(.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$ pip list
Package      Version
-----
pip          20.0.2
pkg-resources 0.0.0
setuptools   44.0.0
(.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$
```

- Instalando dependências: `pip install nome_do_pacote`

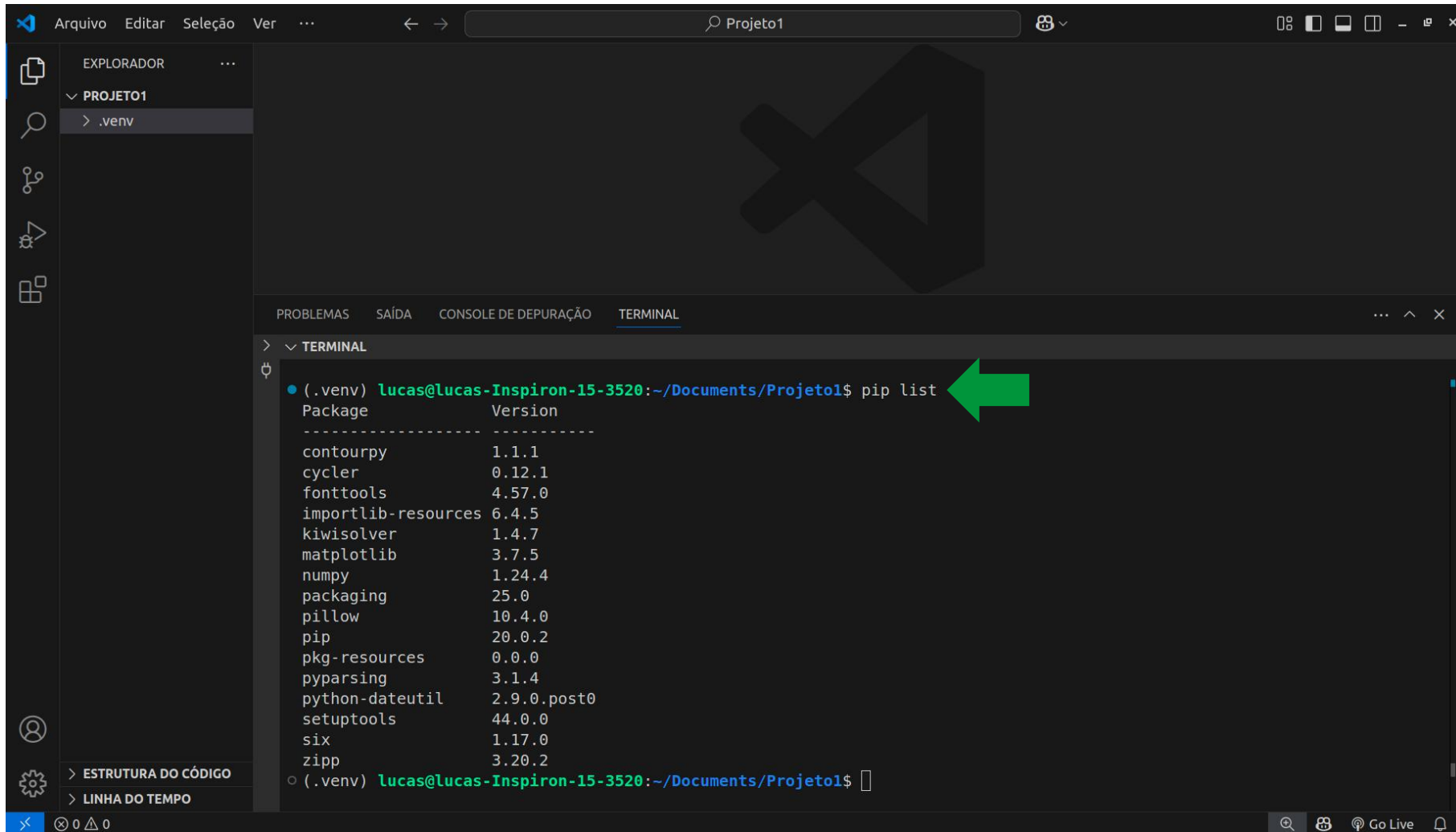


The screenshot shows a code editor interface with a terminal window at the bottom. The terminal displays the command `pip install matplotlib` being executed in a virtual environment. A green arrow points to the command. The output shows the installation progress for matplotlib and its dependencies.

```
> v TERMINAL
• (.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$ pip install matplotlib
Collecting matplotlib
  Using cached matplotlib-3.7.5-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (9.2 MB)
Collecting importlib-resources>=3.2.0; python_version < "3.10"
  Downloading importlib_resources-6.4.5-py3-none-any.whl (36 kB)
Collecting packaging>=20.0
  Downloading packaging-25.0-py3-none-any.whl (66 kB)
  | 66 kB 2.1 MB/s
Collecting pillow>=6.2.0
  Downloading pillow-10.4.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.4 MB)
  | 4.4 MB 11.6 MB/s
Collecting numpy<2,>=1.20
  Using cached numpy-1.24.4-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
```

venv

- Listando as dependências do ambiente virtual: **pip list**



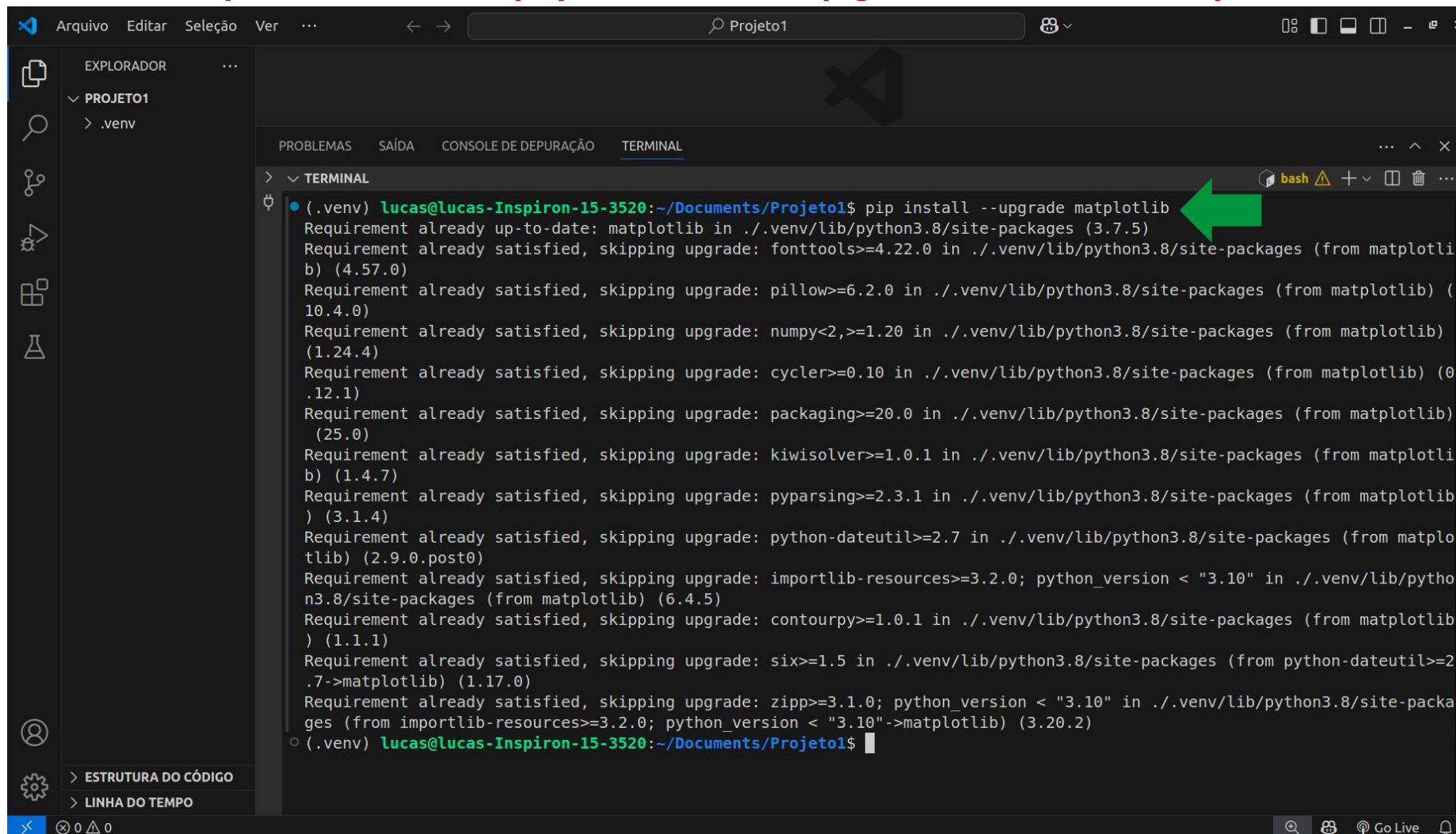
The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal displays the output of the `pip list` command, which lists the installed packages and their versions in the virtual environment. A green arrow points to the command prompt in the terminal.

```
(.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$ pip list
```

Package	Version
contourpy	1.1.1
cycler	0.12.1
fonttools	4.57.0
importlib-resources	6.4.5
kiwisolver	1.4.7
matplotlib	3.7.5
numpy	1.24.4
packaging	25.0
pillow	10.4.0
pip	20.0.2
pkg-resources	0.0.0
pyparsing	3.1.4
python-dateutil	2.9.0.post0
setuptools	44.0.0
six	1.17.0
zipp	3.20.2

```
(.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$
```

- Atualizando dependências: **pip install --upgrade nome-do-pacote**



```
Arquivo  Editar  Seleção  Ver  ...  Projeto1  bash  +  -  x

EXPLORADOR
PROJETO1
.venv

PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL

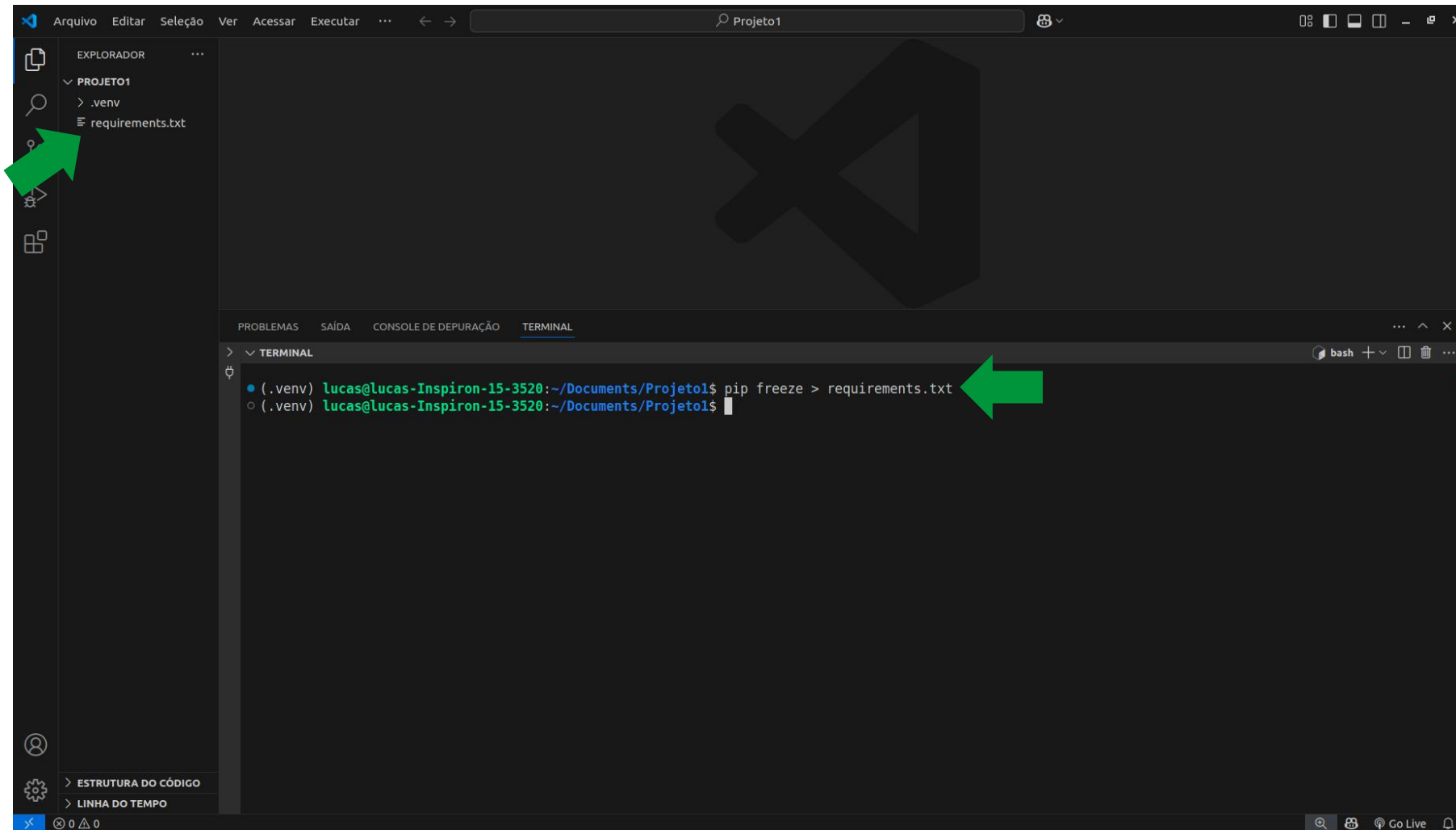
> TERMINAL
(.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$ pip install --upgrade matplotlib
Requirement already up-to-date: matplotlib in ./venv/lib/python3.8/site-packages (3.7.5)
Requirement already satisfied, skipping upgrade: fonttools>=4.22.0 in ./venv/lib/python3.8/site-packages (from matplotlib) (4.57.0)
Requirement already satisfied, skipping upgrade: pillow>=6.2.0 in ./venv/lib/python3.8/site-packages (from matplotlib) (10.4.0)
Requirement already satisfied, skipping upgrade: numpy<2,>=1.20 in ./venv/lib/python3.8/site-packages (from matplotlib) (1.24.4)
Requirement already satisfied, skipping upgrade: cyclor>=0.10 in ./venv/lib/python3.8/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied, skipping upgrade: packaging>=20.0 in ./venv/lib/python3.8/site-packages (from matplotlib) (25.0)
Requirement already satisfied, skipping upgrade: kiwisolver>=1.0.1 in ./venv/lib/python3.8/site-packages (from matplotlib) (1.4.7)
Requirement already satisfied, skipping upgrade: pyparsing>=2.3.1 in ./venv/lib/python3.8/site-packages (from matplotlib) (3.1.4)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.7 in ./venv/lib/python3.8/site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied, skipping upgrade: importlib-resources>=3.2.0; python_version < "3.10" in ./venv/lib/python3.8/site-packages (from matplotlib) (6.4.5)
Requirement already satisfied, skipping upgrade: contourpy>=1.0.1 in ./venv/lib/python3.8/site-packages (from matplotlib) (1.1.1)
Requirement already satisfied, skipping upgrade: six>=1.5 in ./venv/lib/python3.8/site-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Requirement already satisfied, skipping upgrade: zipp>=3.1.0; python_version < "3.10" in ./venv/lib/python3.8/site-packages (from importlib-resources>=3.2.0; python_version < "3.10"->matplotlib) (3.20.2)
(.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$
```


venv

- O requirements.txt é um arquivo de texto que lista todas as bibliotecas (dependências) que um projeto Python precisa para funcionar corretamente.
- Para gerar o arquivo: `pip freeze > requirements.txt`

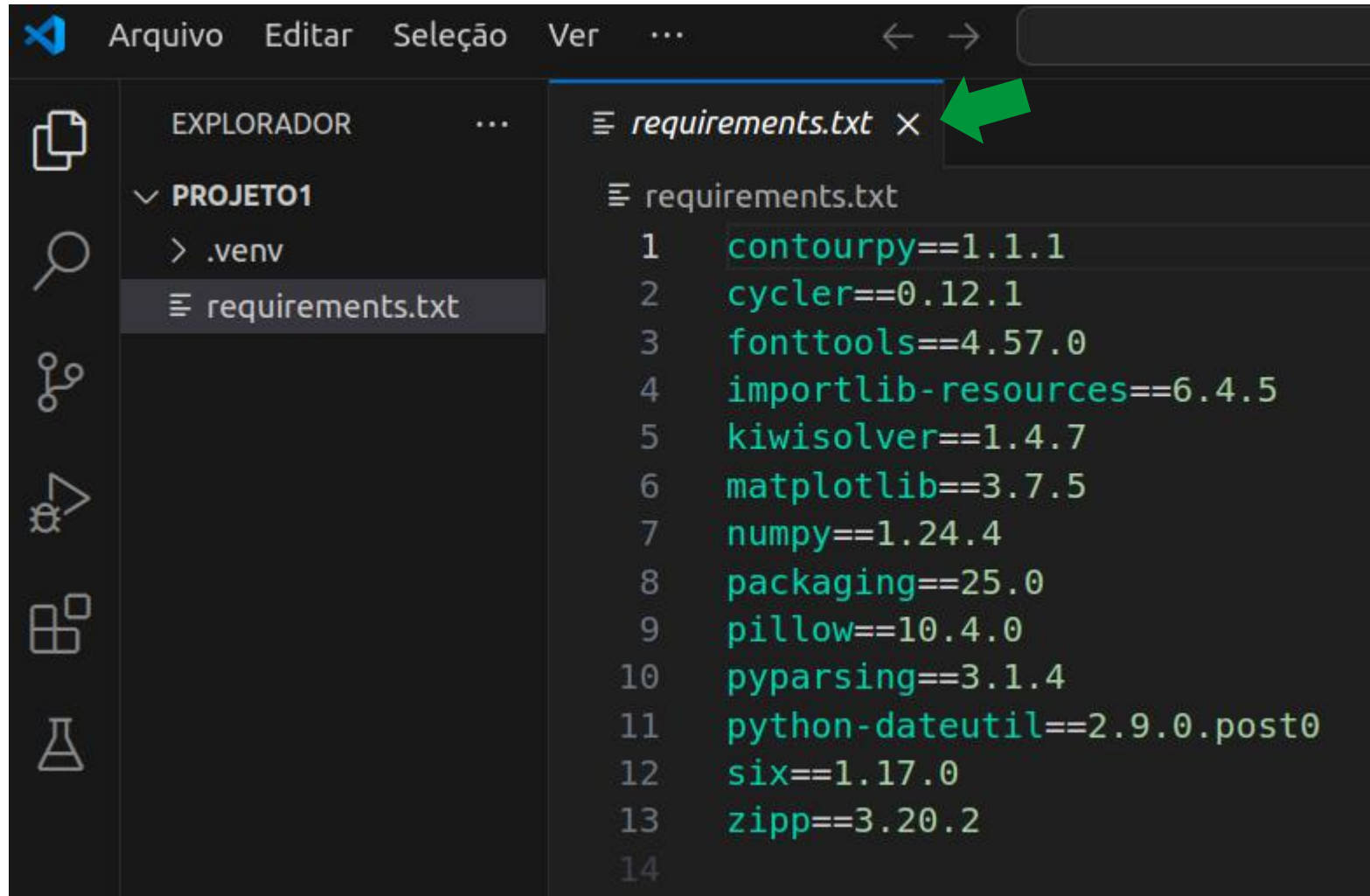
venv

- Para gerar o arquivo: `pip freeze > requirements.txt`



venv

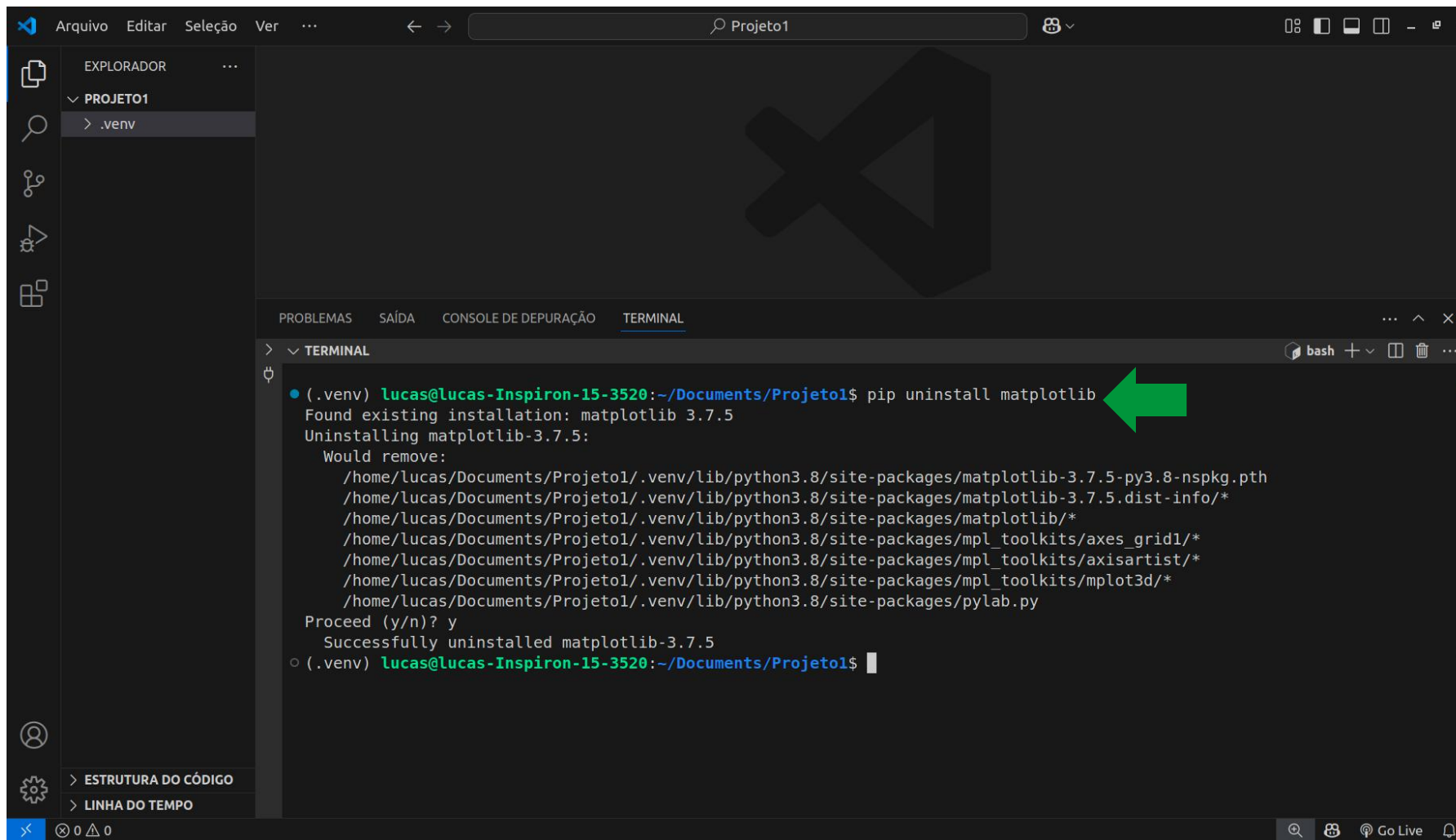
- Para gerar o arquivo: `pip freeze > requirements.txt`



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'PROJETO1' with a subdirectory '.venv' and a file 'requirements.txt'. The main editor area displays the contents of 'requirements.txt', which was generated by the command 'pip freeze > requirements.txt'. A green arrow points to the file name 'requirements.txt' in the tab bar at the top of the editor.

```
requirements.txt
1  contourpy==1.1.1
2  cycler==0.12.1
3  fonttools==4.57.0
4  importlib-resources==6.4.5
5  kiwisolver==1.4.7
6  matplotlib==3.7.5
7  numpy==1.24.4
8  packaging==25.0
9  pillow==10.4.0
10 pyparsing==3.1.4
11 python-dateutil==2.9.0.post0
12 six==1.17.0
13 zipp==3.20.2
14
```

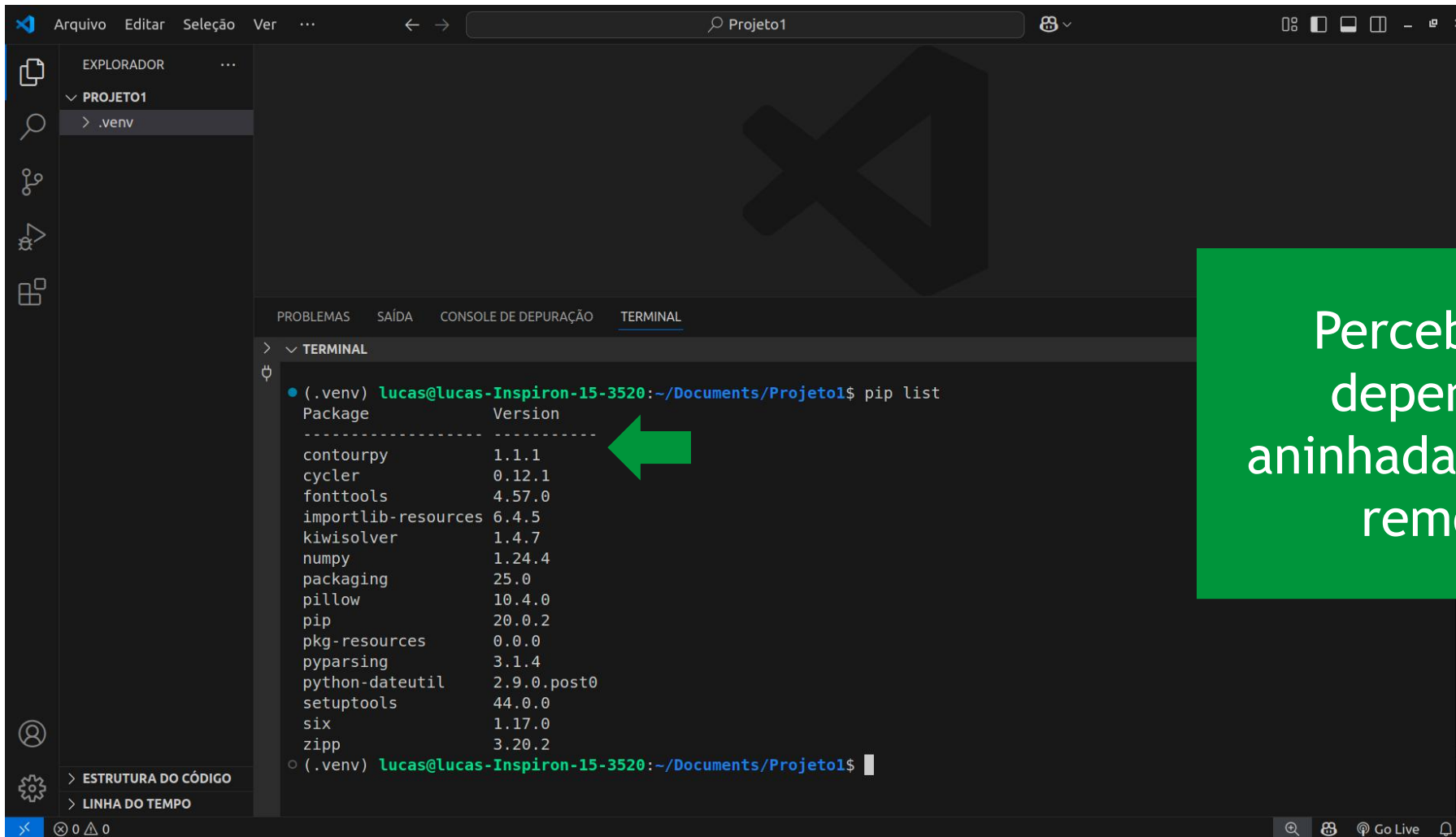
- Removendo dependências do ambiente virtual: **pip uninstall nome_do_pacote**



```
Arquivo  Editar  Seleção  Ver  ...  Projeto1  bash +  ...  
EXPLORADOR  
PROJETO1  
  .venv  
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  
> TERMINAL  
• (.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$ pip uninstall matplotlib  
Found existing installation: matplotlib 3.7.5  
Uninstalling matplotlib-3.7.5:  
Would remove:  
  /home/lucas/Documents/Projeto1/.venv/lib/python3.8/site-packages/matplotlib-3.7.5-py3.8-nspkg.pth  
  /home/lucas/Documents/Projeto1/.venv/lib/python3.8/site-packages/matplotlib-3.7.5.dist-info/*  
  /home/lucas/Documents/Projeto1/.venv/lib/python3.8/site-packages/matplotlib/*  
  /home/lucas/Documents/Projeto1/.venv/lib/python3.8/site-packages/mpl_toolkits/axes_grid1/*  
  /home/lucas/Documents/Projeto1/.venv/lib/python3.8/site-packages/mpl_toolkits/axisartist/*  
  /home/lucas/Documents/Projeto1/.venv/lib/python3.8/site-packages/mpl_toolkits/mplot3d/*  
  /home/lucas/Documents/Projeto1/.venv/lib/python3.8/site-packages/pylab.py  
Proceed (y/n)? y  
Successfully uninstalled matplotlib-3.7.5  
• (.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$  
> ESTRUTURA DO CÓDIGO  
> LINHA DO TEMPO
```

venv

- Listando as dependências do ambiente virtual: **pip list**



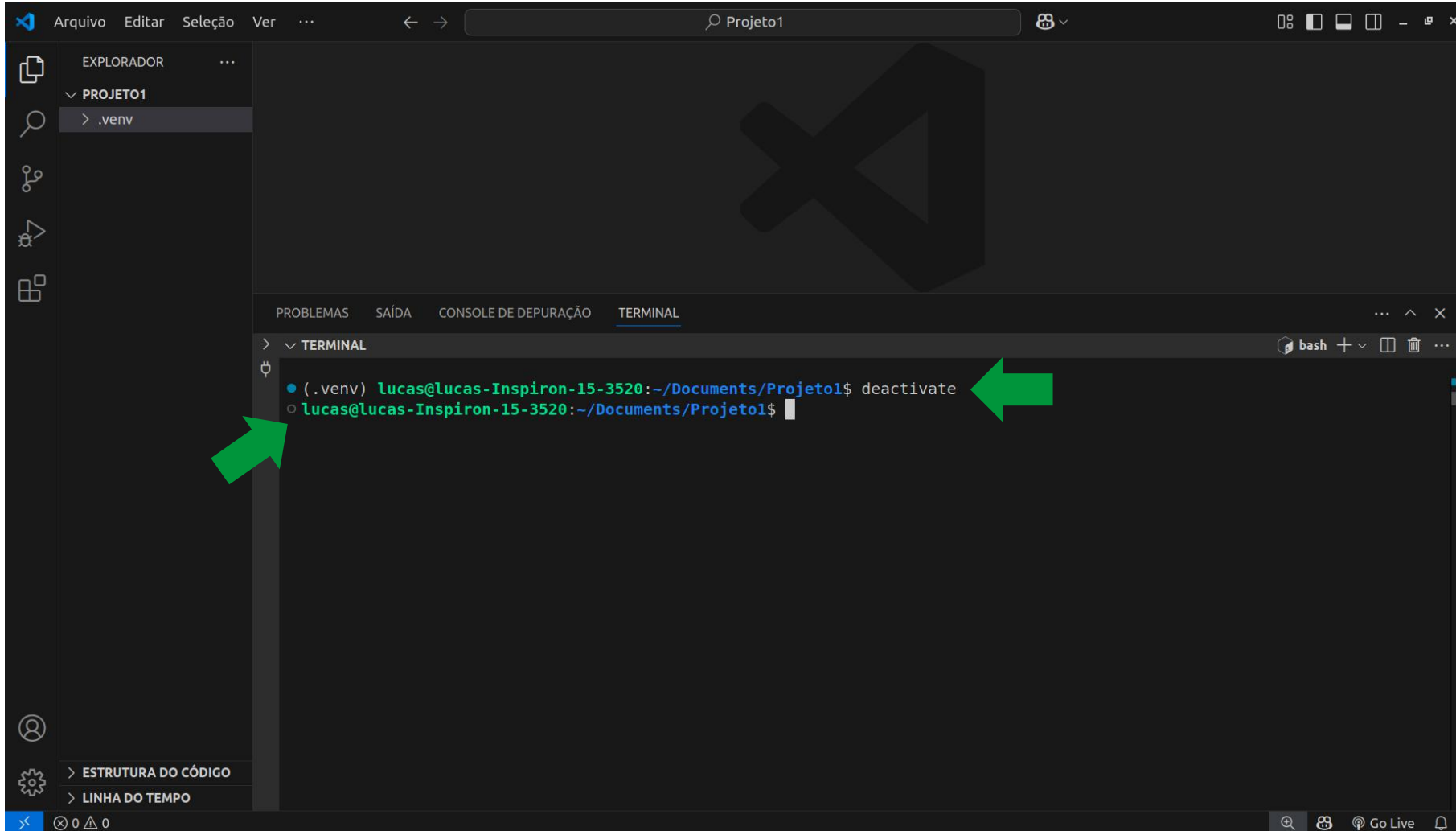
```
Arquivo  Editar  Seleção  Ver  ...  Projeto1  [Icons]
EXPLORADOR
  PROJETO1
    .venv

PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL
> TERMINAL
(.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$ pip list
Package            Version
-----
contourpy           1.1.1
cycler               0.12.1
fonttools           4.57.0
importlib-resources 6.4.5
kiwisolver           1.4.7
numpy                1.24.4
packaging            25.0
pillow               10.4.0
pip                  20.0.2
pkg-resources        0.0.0
pyparsing            3.1.4
python-dateutil      2.9.0.post0
setuptools           44.0.0
six                  1.17.0
zip                  3.20.2
(.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$
```

Perceba que as dependências aninhadas não foram removidas.

venv

- Desativando o ambiente virtual: **deactivate**



The screenshot shows the Visual Studio Code interface with a dark theme. The Explorer sidebar on the left shows a project named 'PROJETO1' with a subdirectory '.venv'. The Terminal panel at the bottom is active, showing a bash shell. The command prompt is '(.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1\$'. The command 'deactivate' has been entered and is highlighted by a green arrow. Another green arrow points to the terminal output area below the command.

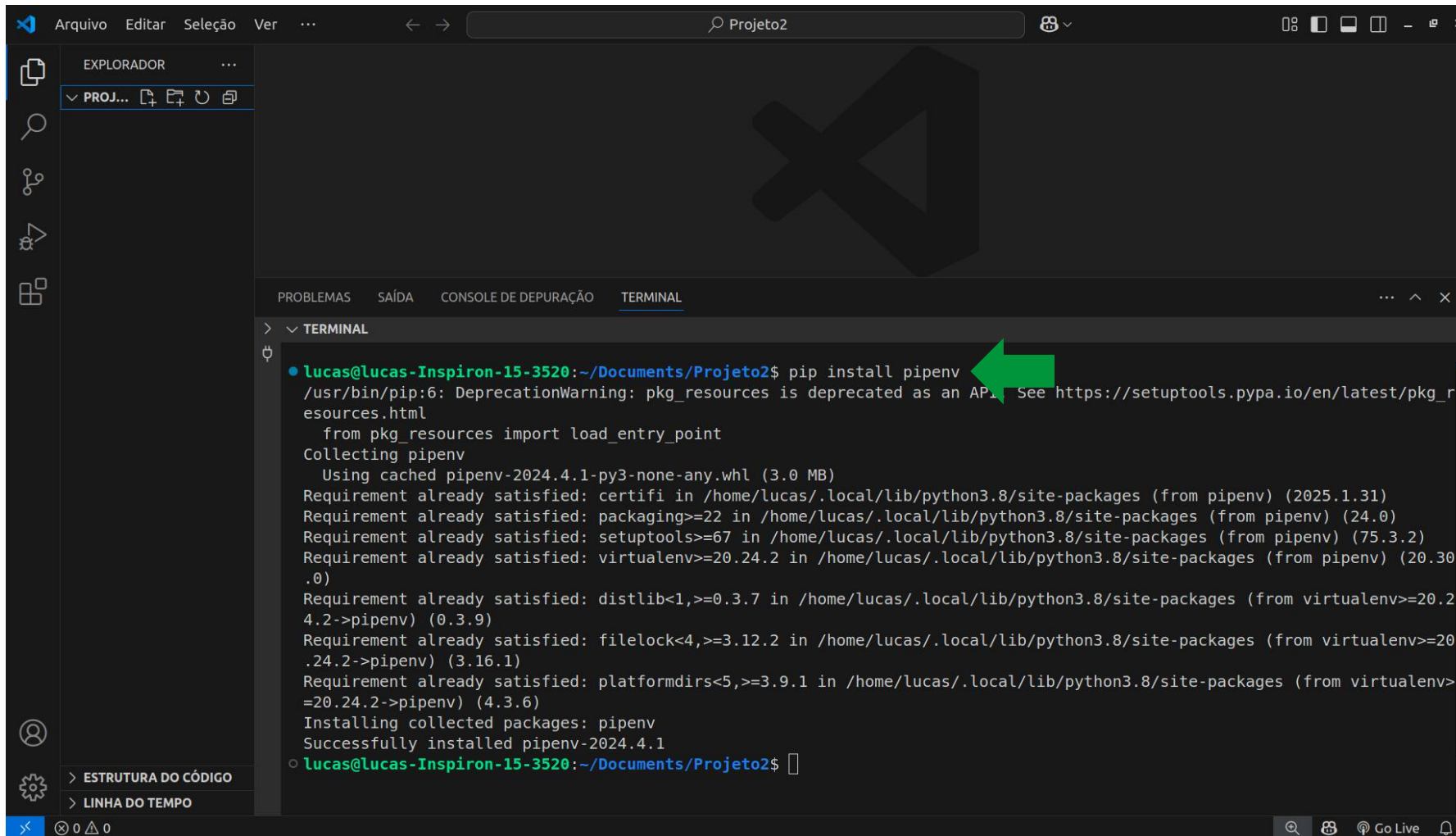
```
> v TERMINAL
❏ bash + ▾ □ ✕ ...
❏ (.venv) lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$ deactivate
❏ lucas@lucas-Inspiron-15-3520:~/Documents/Projeto1$
```

pipenv

- O pipenv é uma ferramenta que combina e melhora o uso do pip e do venv em um só comando. Ele foi criado para facilitar a gestão de dependências e ambientes virtuais em projetos Python.
- O pipenv adiciona/remove pacotes automaticamente do arquivo Pipfile conforme você instala e desinstala pacotes.
 - O Pipfile é um arquivo criado pelo pipenv que substitui o requirements.txt.

pipenv

- Instalando o pipenv (ambiente global): **pip install pipenv**



Arquivo Editar Seleção Ver ... Projeto2

EXPLORADOR

PROJ...

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL

TERMINAL

```
lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$ pip install pipenv
/usr/bin/pip:6: DeprecationWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
  from pkg_resources import load_entry_point
Collecting pipenv
  Using cached pipenv-2024.4.1-py3-none-any.whl (3.0 MB)
Requirement already satisfied: certifi in /home/lucas/.local/lib/python3.8/site-packages (from pipenv) (2025.1.31)
Requirement already satisfied: packaging>=22 in /home/lucas/.local/lib/python3.8/site-packages (from pipenv) (24.0)
Requirement already satisfied: setuptools>=67 in /home/lucas/.local/lib/python3.8/site-packages (from pipenv) (75.3.2)
Requirement already satisfied: virtualenv>=20.24.2 in /home/lucas/.local/lib/python3.8/site-packages (from pipenv) (20.30.0)
Requirement already satisfied: distlib<1,>=0.3.7 in /home/lucas/.local/lib/python3.8/site-packages (from virtualenv>=20.24.2->pipenv) (0.3.9)
Requirement already satisfied: filelock<4,>=3.12.2 in /home/lucas/.local/lib/python3.8/site-packages (from virtualenv>=20.24.2->pipenv) (3.16.1)
Requirement already satisfied: platformdirs<5,>=3.9.1 in /home/lucas/.local/lib/python3.8/site-packages (from virtualenv>=20.24.2->pipenv) (4.3.6)
Installing collected packages: pipenv
Successfully installed pipenv-2024.4.1
lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$
```

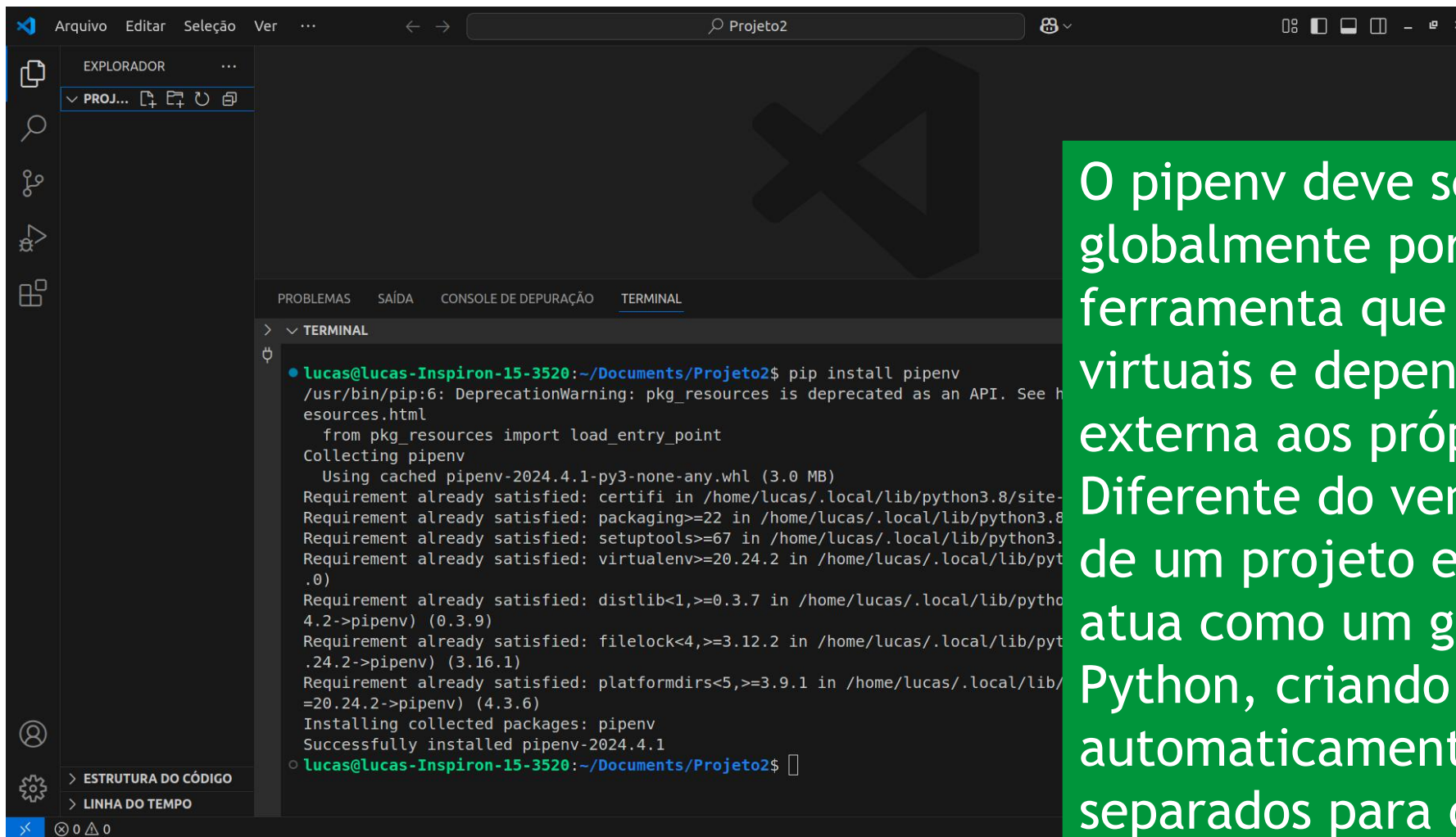
ESTRUTURA DO CÓDIGO

LINHA DO TEMPO

0 0 0

Go Live

- Instalando o pipenv (ambiente global): **pip install pipenv**

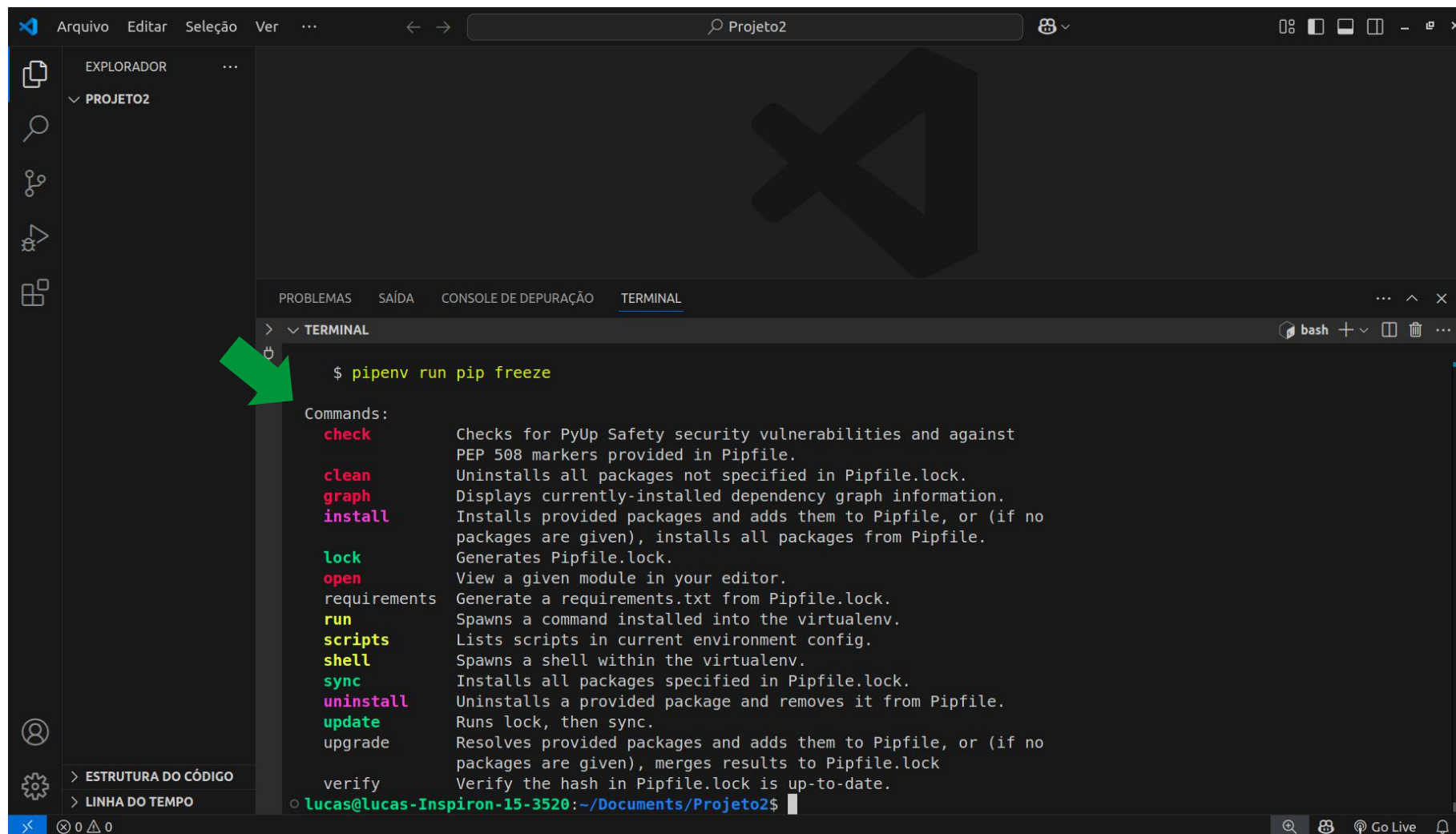


```
Arquivo  Editar  Seleção  Ver  ...  Projeto2
EXPLORADOR
PROJ...
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL
> v TERMINAL
• lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$ pip install pipenv
/usr/bin/pip:6: DeprecationWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
  from pkg_resources import load_entry_point
Collecting pipenv
  Using cached pipenv-2024.4.1-py3-none-any.whl (3.0 MB)
Requirement already satisfied: certifi in /home/lucas/.local/lib/python3.8/site-packages (2024.2.2)
Requirement already satisfied: packaging>=22 in /home/lucas/.local/lib/python3.8/site-packages (24.0)
Requirement already satisfied: setuptools>=67 in /home/lucas/.local/lib/python3.8/site-packages (67.7.2)
Requirement already satisfied: virtualenv>=20.24.2 in /home/lucas/.local/lib/python3.8/site-packages (20.24.2)
Requirement already satisfied: distlib<1, >=0.3.7 in /home/lucas/.local/lib/python3.8/site-packages (0.3.9)
Requirement already satisfied: filelock<4, >=3.12.2 in /home/lucas/.local/lib/python3.8/site-packages (3.16.1)
Requirement already satisfied: platformdirs<5, >=3.9.1 in /home/lucas/.local/lib/python3.8/site-packages (4.3.6)
Installing collected packages: pipenv
Successfully installed pipenv-2024.4.1
• lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$
```

O pipenv deve ser instalado globalmente porque ele é uma ferramenta que gerencia ambientes virtuais e dependências de forma externa aos próprios ambientes. Diferente do venv, que é usado dentro de um projeto específico, o pipenv atua como um gerenciador de projetos Python, criando e organizando automaticamente ambientes virtuais separados para cada aplicação.

pipenv

- Conhecendo os comandos do pipenv: **pipenv**



Arquivo Editar Seleção Ver ... Projeto2

EXPLORADOR

PROJETO2

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO **TERMINAL**

bash

```
$ pipenv run pip freeze
```

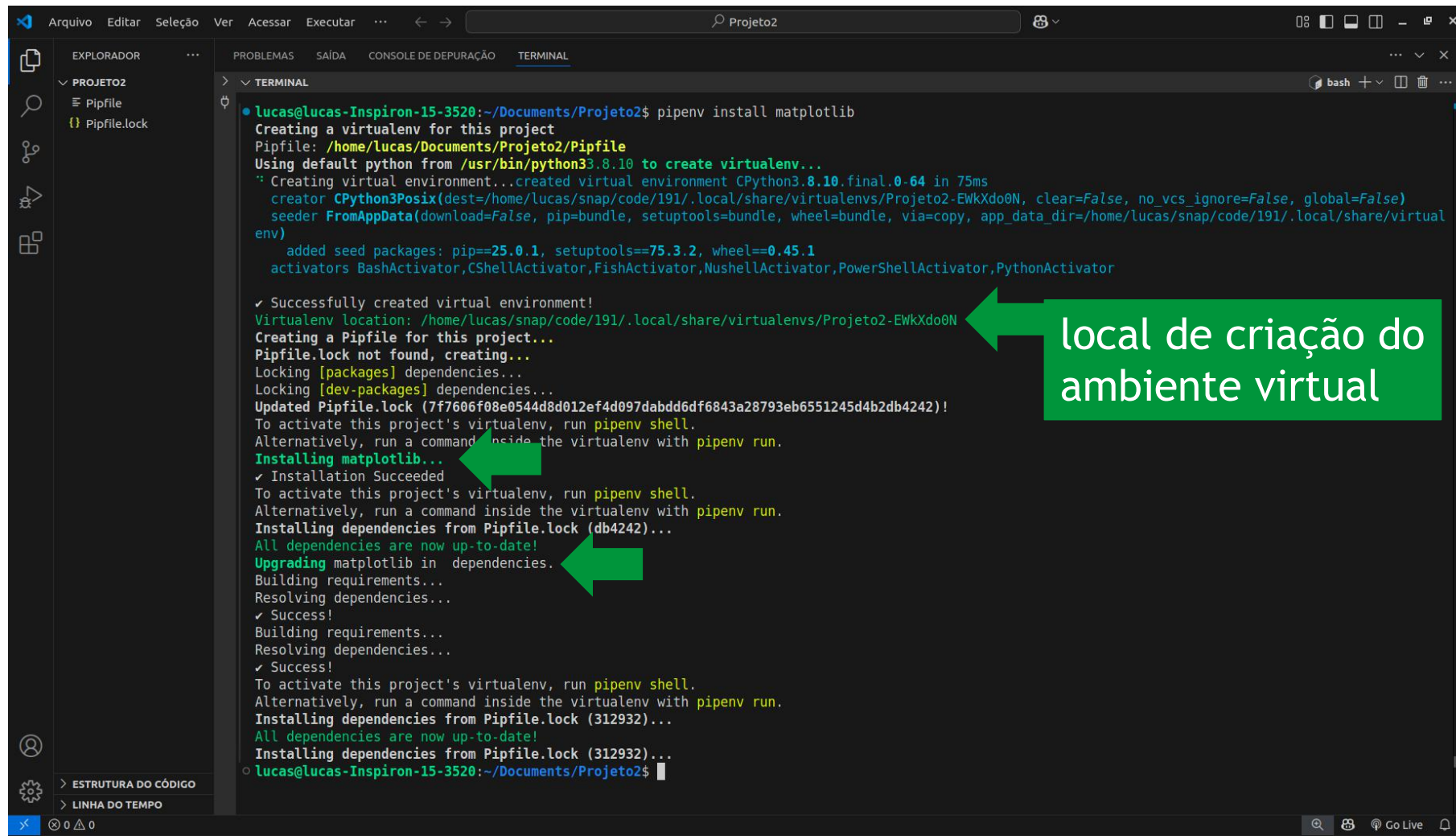
Commands:

check	Checks for PyUp Safety security vulnerabilities and against PEP 508 markers provided in Pipfile.
clean	Uninstalls all packages not specified in Pipfile.lock.
graph	Displays currently-installed dependency graph information.
install	Installs provided packages and adds them to Pipfile, or (if no packages are given), installs all packages from Pipfile.
lock	Generates Pipfile.lock.
open	View a given module in your editor.
requirements	Generate a requirements.txt from Pipfile.lock.
run	Spawns a command installed into the virtualenv.
scripts	Lists scripts in current environment config.
shell	Spawns a shell within the virtualenv.
sync	Installs all packages specified in Pipfile.lock.
uninstall	Uninstalls a provided package and removes it from Pipfile.
update	Runs lock, then sync.
upgrade	Resolves provided packages and adds them to Pipfile, or (if no packages are given), merges results to Pipfile.lock
verify	Verify the hash in Pipfile.lock is up-to-date.

lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2\$

pipenv

- Criando um ambiente virtual e instalando uma dependência: **pipenv install nome_da_dependencia**



```
Arquivo  Editar  Seleção  Ver  Acessar  Executar  ...  ←  →  Projeto2
EXPLORADOR  ...  PROBLEMAS  SAÍDA  CONSOLE DE DEPUÇÃO  TERMINAL
PROJETO2
  Pipfile
  Pipfile.lock
TERMINAL
lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$ pipenv install matplotlib
Creating a virtualenv for this project
Pipfile: /home/lucas/Documents/Projeto2/Pipfile
Using default python from /usr/bin/python3.8.10 to create virtualenv...
* Creating virtual environment...created virtual environment CPython3.8.10.final.0-64 in 75ms
  creator CPython3Posix(dest=/home/lucas/snap/code/191/.local/share/virtualenvs/Projeto2-EwkXdo0N, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/lucas/snap/code/191/.local/share/virtual
env)
    added seed packages: pip==25.0.1, setuptools==75.3.2, wheel==0.45.1
    activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

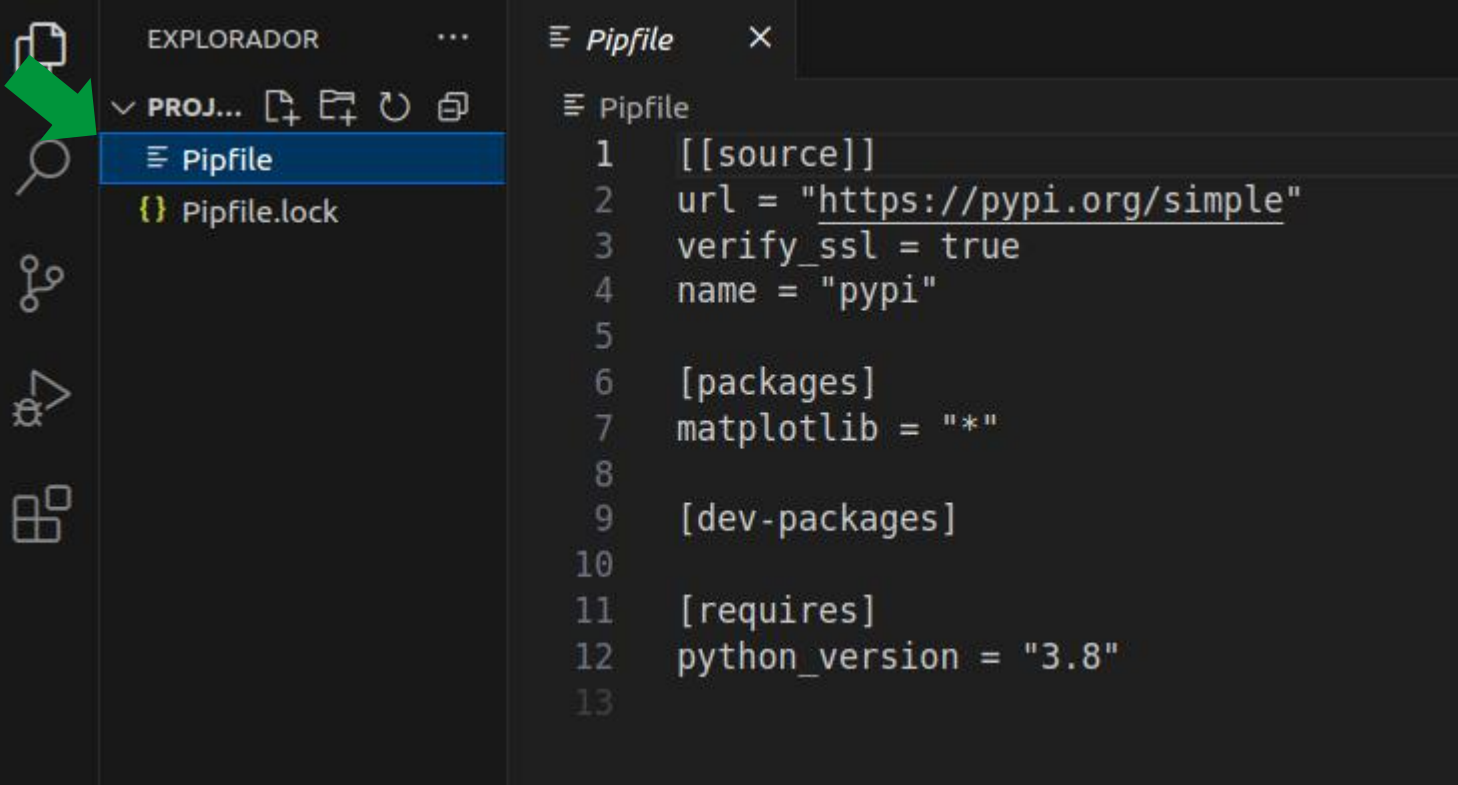
✓ Successfully created virtual environment!
Virtualenv location: /home/lucas/snap/code/191/.local/share/virtualenvs/Projeto2-EwkXdo0N
Creating a Pipfile for this project...
Pipfile.lock not found, creating...
Locking [packages] dependencies...
Locking [dev-packages] dependencies...
Updated Pipfile.lock (7f7606f08e0544d8d012ef4d097dabdd6df6843a28793eb6551245d4b2db4242)!
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing matplotlib...
✓ Installation Succeeded
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (db4242)...
All dependencies are now up-to-date!
Upgrading matplotlib in dependencies.
Building requirements...
Resolving dependencies...
✓ Success!
Building requirements...
Resolving dependencies...
✓ Success!
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (312932)...
All dependencies are now up-to-date!
Installing dependencies from Pipfile.lock (312932)...
lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$
```

local de criação do ambiente virtual

pipenv

- Visualizando os arquivos criados no diretório:

O Pipfile é um arquivo criado pelo pipenv que substitui o antigo requirements.txt como forma moderna e organizada de gerenciar as dependências de um projeto Python.



```
EXPLORADOR
PROJ...
Pipfile
Pipfile.lock

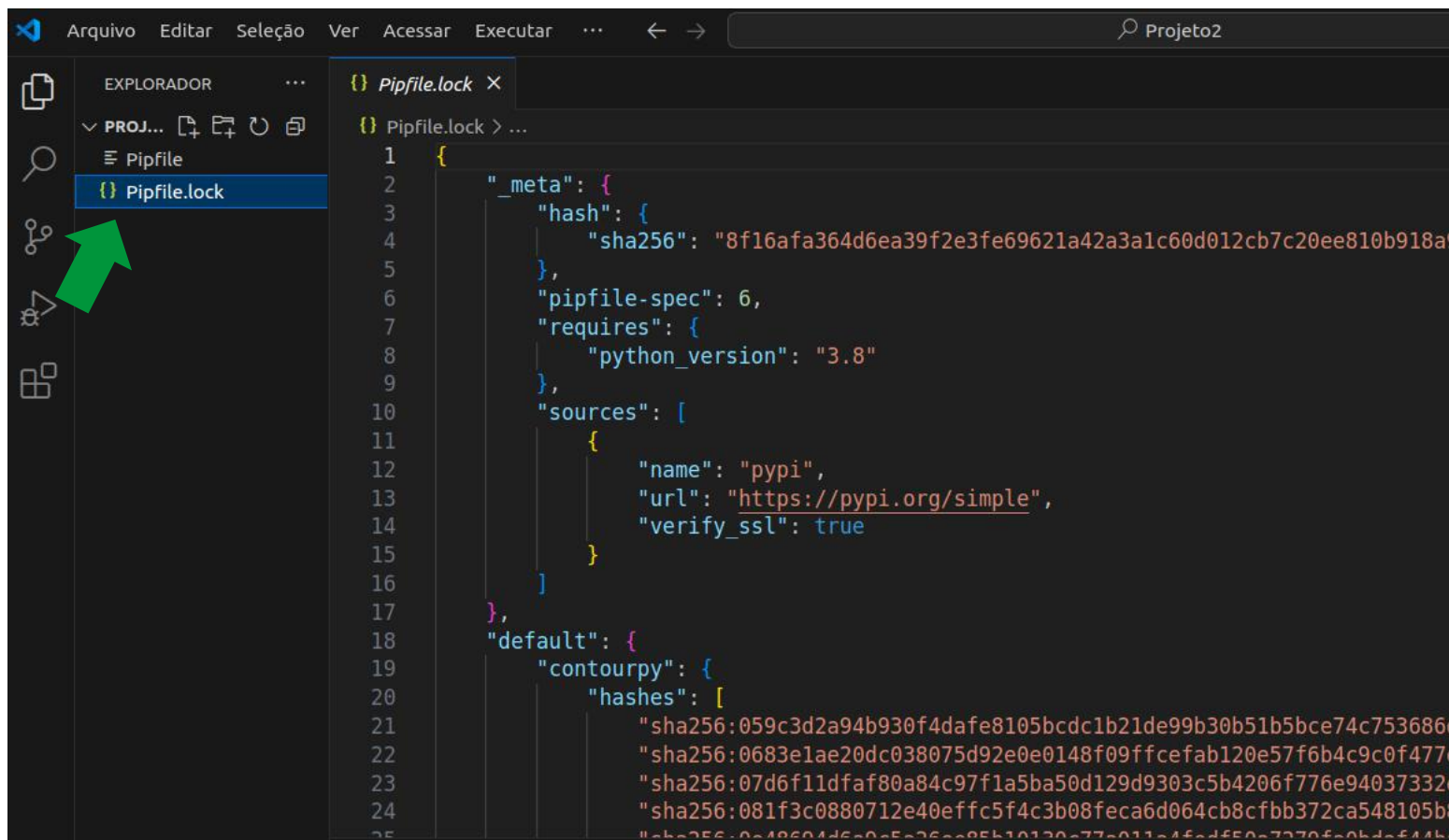
Pipfile
1  [[source]]
2  url = "https://pypi.org/simple"
3  verify_ssl = true
4  name = "pypi"
5
6  [packages]
7  matplotlib = "*"
8
9  [dev-packages]
10
11 [requires]
12 python_version = "3.8"
13
```


pipenv

- Visualizando os arquivos criados no diretório:

Quando pacotes são instalados com o pipenv, ele atualiza o Pipfile.lock, que:

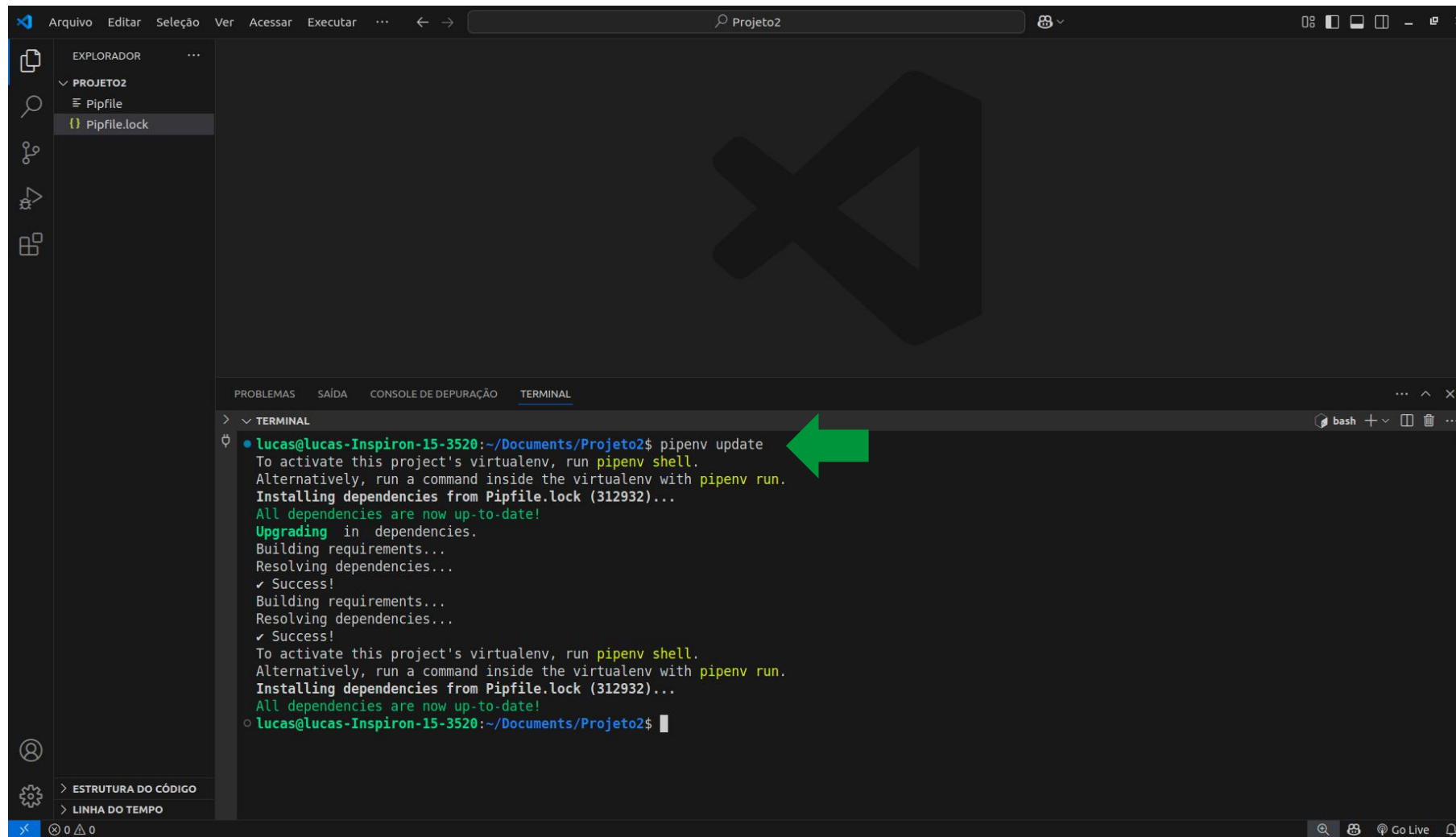
- Trava as versões exatas das dependências e subdependências.
- Garante que o ambiente será reproduzido de forma idêntica em qualquer máquina.



```
Pipfile.lock
{
  "_meta": {
    "hash": {
      "sha256": "8f16afa364d6ea39f2e3fe69621a42a3a1c60d012cb7c20ee810b918a..."
    },
    "pipfile-spec": 6,
    "requires": {
      "python_version": "3.8"
    },
    "sources": [
      {
        "name": "pypi",
        "url": "https://pypi.org/simple",
        "verify_ssl": true
      }
    ]
  },
  "default": {
    "contourpy": {
      "hashes": [
        "sha256:059c3d2a94b930f4dafa8105bcd1b21de99b30b51b5bce74c753686...",
        "sha256:0683e1ae20dc038075d92e0e0148f09ffcefab120e57f6b4c9c0f477...",
        "sha256:07d6f11dfaf80a84c97f1a5ba50d129d9303c5b4206f776e94037332...",
        "sha256:081f3c0880712e40effc5f4c3b08feca6d064cb8cfbb372ca548105b...",
        "sha256:0c48604d6c0c5c26c005b10130c77c011c4fcd550c7370f0bde-f44b..."
      ]
    }
  }
}
```

pipenv

- Atualizando dependências: **pipenv update**



```
Arquivo  Editar  Seleção  Ver  Acessar  Executar  ...  Projeto2  bash +  Go Live  🔔
```

EXPLORADOR

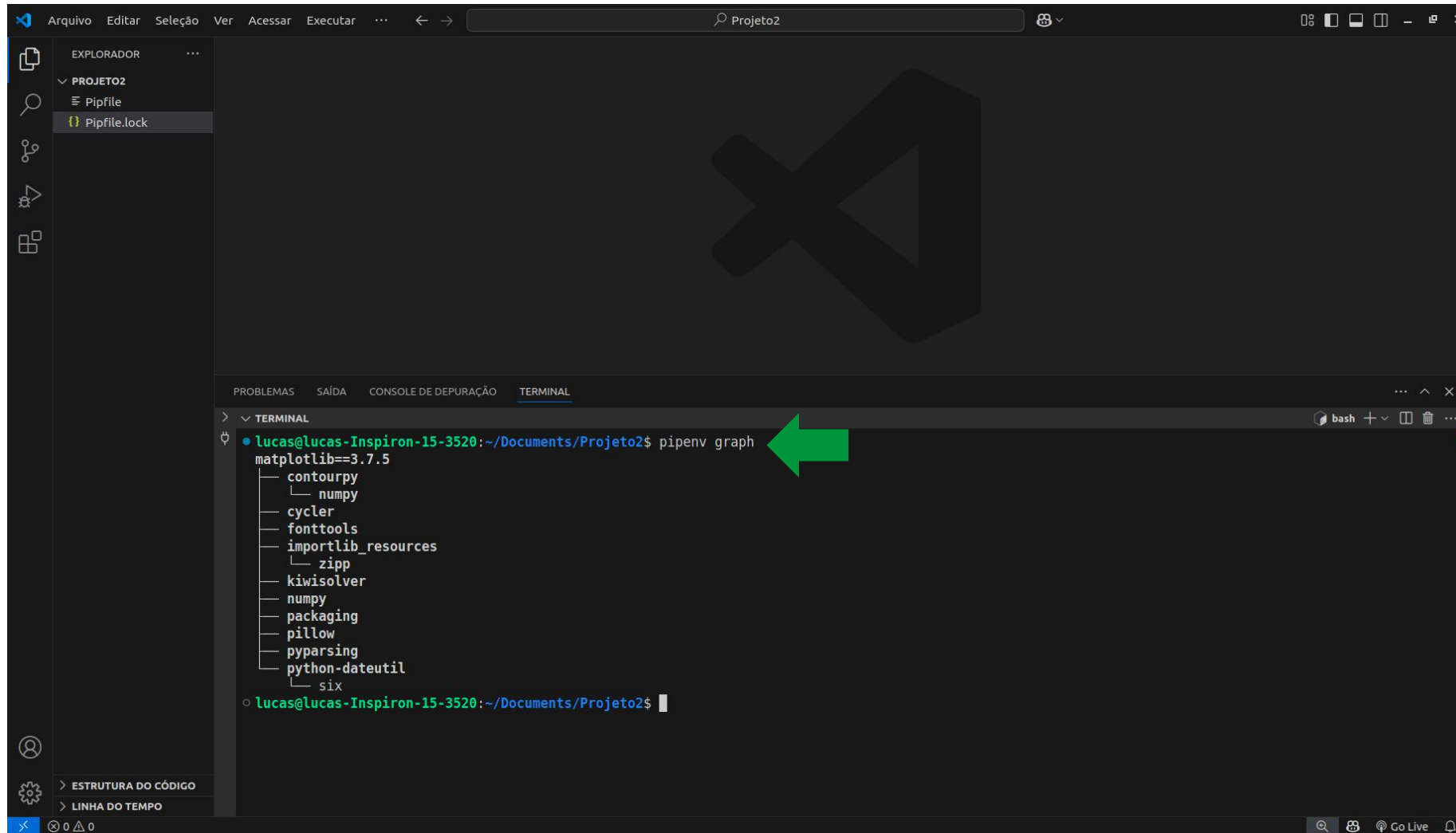
- PROJETO2
 - Pipfile
 - Pipfile.lock

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO **TERMINAL**

```
> TERMINAL
• lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$ pipenv update
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (312932)...
All dependencies are now up-to-date!
Upgrading in dependencies.
Building requirements...
Resolving dependencies...
✓ Success!
Building requirements...
Resolving dependencies...
✓ Success!
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (312932)...
All dependencies are now up-to-date!
o lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$
```

pipenv

- Visualizar a árvore (graph) de dependências: **pipenv graph**

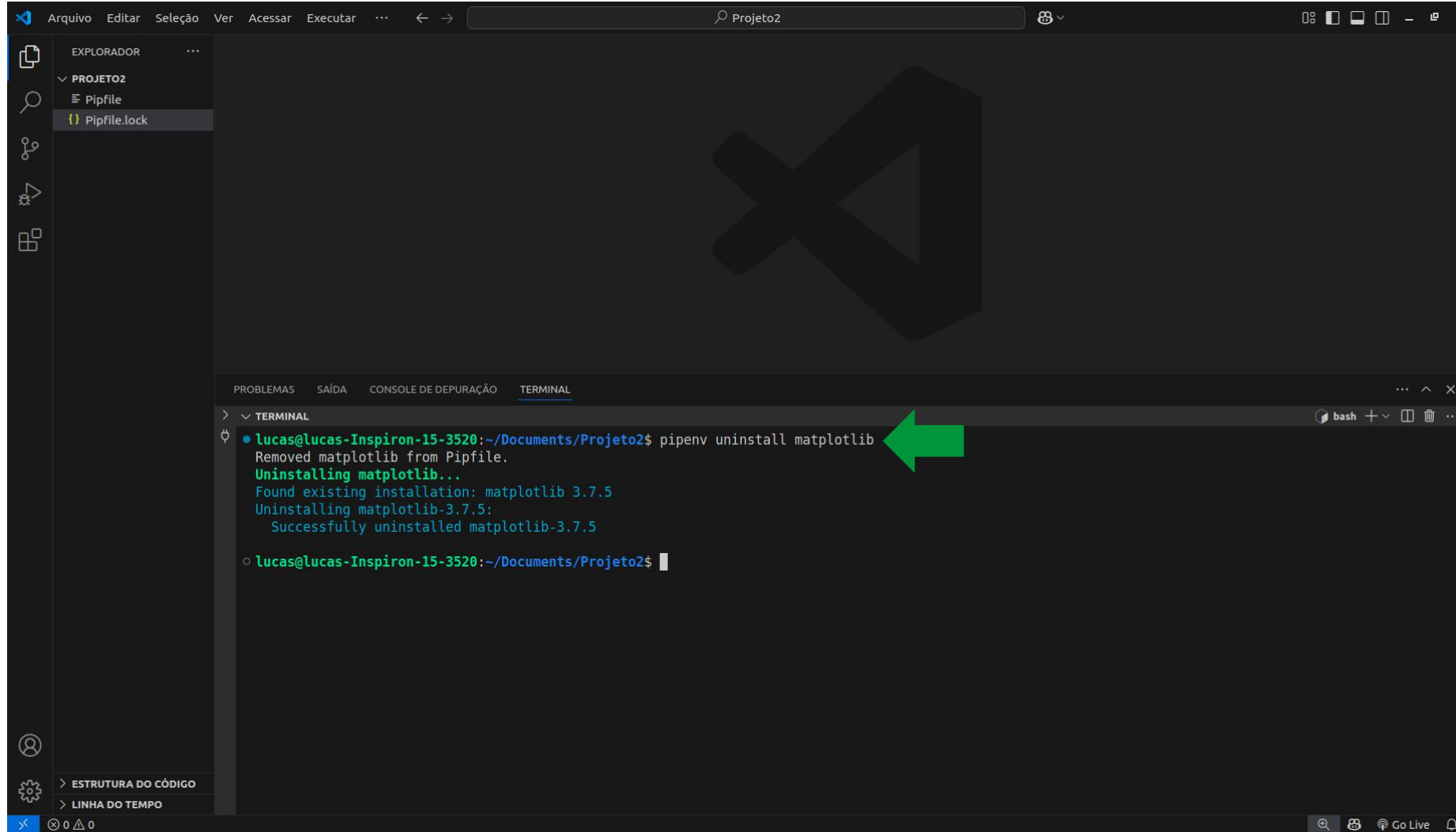


The screenshot shows the Visual Studio Code interface with a project named 'Projeto2'. The Explorer sidebar on the left shows the file structure with 'Pipfile' and 'Pipfile.lock'. The Terminal panel at the bottom displays the command `pipenv graph` being executed. A green arrow points to the command. The output shows a dependency tree for `matplotlib==3.7.5`, listing its sub-dependencies.

```
• lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$ pipenv graph
matplotlib==3.7.5
├── contourpy
│   └── numpy
├── cyclor
├── fonttools
├── importlib_resources
│   └── zipp
├── kiwisolver
├── numpy
├── packaging
├── pillow
├── pyparsing
├── python-dateutil
│   └── six
└── o lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$
```

pipenv

- Removendo dependências: `pipenv uninstall nome_do_pacote`



The screenshot shows the Visual Studio Code editor with a project named 'Projeto2'. The Explorer sidebar on the left shows the file structure with 'Pipfile' and 'Pipfile.lock'. The Terminal panel at the bottom displays the following output:

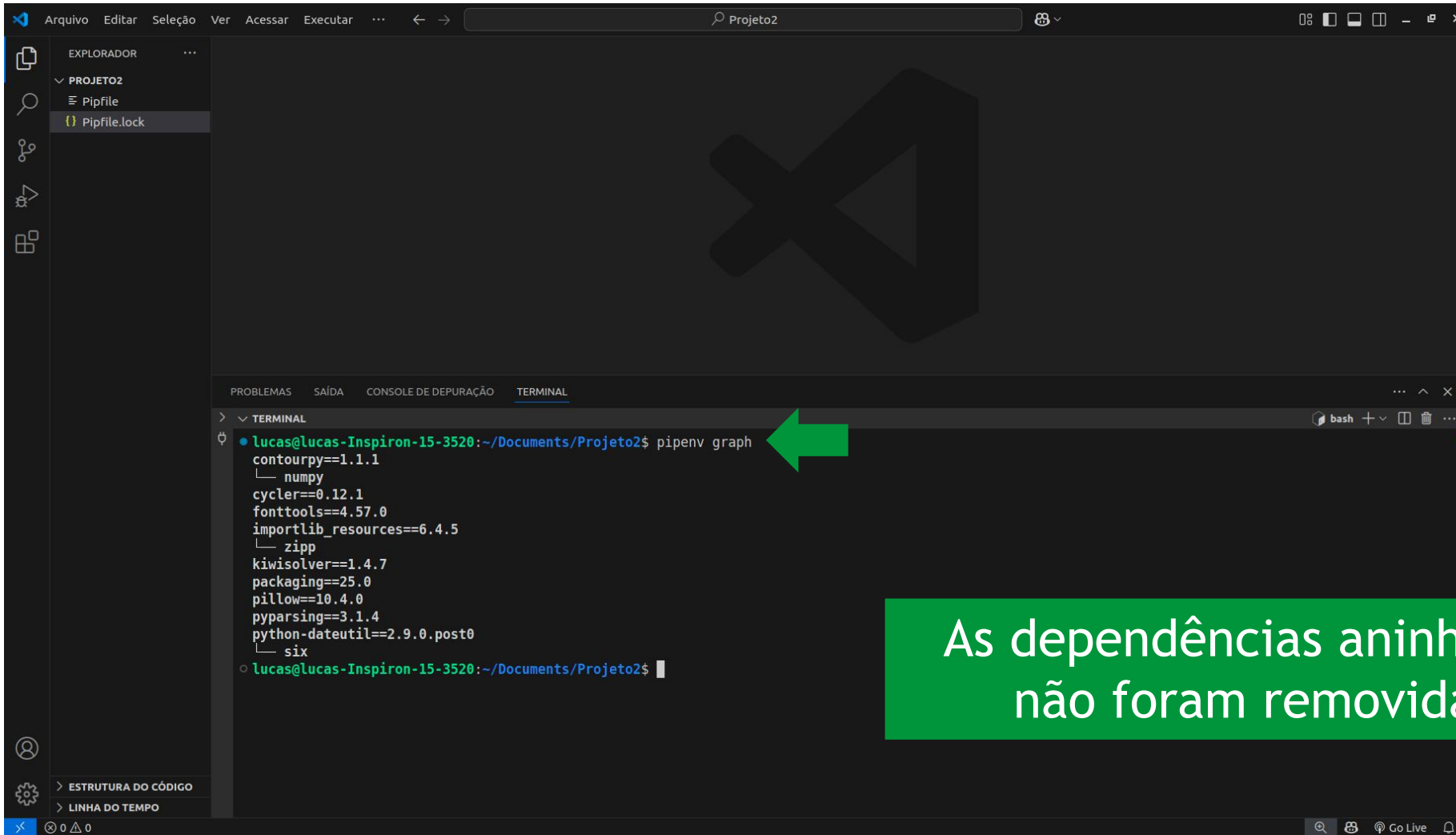
```
lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$ pipenv uninstall matplotlib
Removed matplotlib from Pipfile.
Uninstalling matplotlib...
Found existing installation: matplotlib 3.7.5
Uninstalling matplotlib-3.7.5:
  Successfully uninstalled matplotlib-3.7.5

lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$
```

A green arrow points to the command `pipenv uninstall matplotlib` in the terminal.

pipenv

- Visualizar a árvore (graph) de dependências: **pipenv graph**



```
Arquivo  Editar  Seleção  Ver  Acessar  Executar  ...  ←  →  Projeto2  [Icons]

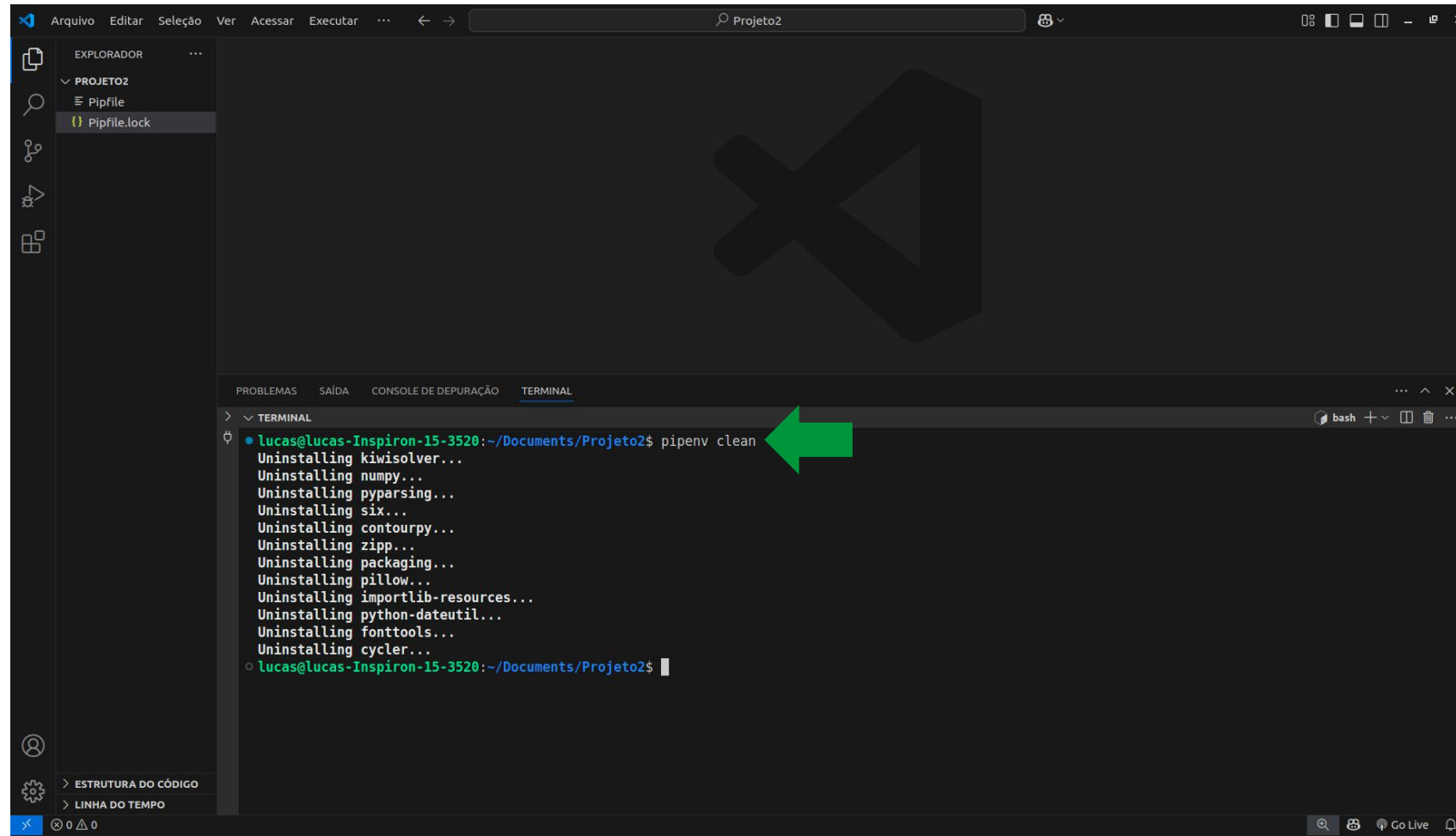
EXPLORADOR
  PROJETO2
    Pipfile
    Pipfile.lock

PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL
  lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$ pipenv graph
  contourpy==1.1.1
  └─ numpy
  cyclar==0.12.1
  fonttools==4.57.0
  importlib_resources==6.4.5
  └─ zipp
  kiwisolver==1.4.7
  packaging==25.0
  pillow==10.4.0
  pyparsing==3.1.4
  python-dateutil==2.9.0.post0
  └─ six
  o lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$
```

As dependências aninhadas não foram removidas

pipenv

- pipenv clean: remove pacotes que estão instalados no ambiente virtual, mas que não estão listados no Pipfile nem no Pipfile.lock



```
Arquivo  Editar  Seleção  Ver  Acessar  Executar  ...  Projeto2  bash +  ...  x
```

EXPLORADOR

- PROJETO2
 - Pipfile
 - Pipfile.lock

PROBLEMAS SAÍDA CONSOLE DE DEPUÇÃO TERMINAL

TERMINAL

```
lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$ pipenv clean
Uninstalling kiwisolver...
Uninstalling numpy...
Uninstalling pyparsing...
Uninstalling six...
Uninstalling contourpy...
Uninstalling zipp...
Uninstalling packaging...
Uninstalling pillow...
Uninstalling importlib-resources...
Uninstalling python-dateutil...
Uninstalling fonttools...
Uninstalling cycler...
lucas@lucas-Inspiron-15-3520:~/Documents/Projeto2$
```

- Mais sobre o pipenv: <https://pipenv.pypa.io/en/latest/>



Star 25,034

Search the doc

Pipenv is a production-ready tool that aims to bring the best of all packaging worlds to the Python world. It harnesses Pipfile, pip, and virtualenv into one single command.

It features very pretty terminal colors.

Stay Informed

Receive updates on new releases and upcoming projects.

Follow @pypa 1,889

Follow @ThePyPA

Join Mailing List.

Other Projects

Pipenv: Python Dev Workflow for Humans

pypi v2024.4.1 license MIT License (MIT) python 3.8 | 3.9 | 3.10 | 3.11 | 3.12 | 3.13

Pipenv is a Python virtualenv management tool that supports a multitude of systems and nicely bridges the gaps between pip, python (using system python, pyenv or asdf) and virtualenv. *Linux, macOS, and Windows are all first-class citizens in pipenv.*

Pipenv automatically creates and manages a virtualenv for your projects, as well as adds/removes packages from your **Pipfile** as you install/uninstall packages. It also generates a project **Pipfile.lock**, which is used to produce deterministic builds.

Pipenv is primarily meant to provide users and developers of applications with an easy method to arrive at a consistent working project environment.

The problems that Pipenv seeks to solve are multi-faceted:

- You no longer need to use **pip** and **virtualenv** separately: they work together.
- Managing a **requirements.txt** file with package hashes can be problematic. Pipenv uses **Pipfile** and **Pipfile.lock** to separate abstract dependency declarations from the last tested combination.
- Hashes are documented in the lock file which are verified during install. Security considerations are put first.
- Strongly encourage the use of the latest versions of dependencies to minimize security risks arising from outdated components.
- Gives you insight into your dependency graph (e.g. `$ pipenv graph`).
- Streamline development workflow by supporting local customizations with **.env** files.

Install Pipenv Today!

The recommended way to install pipenv on most platforms is to install from pypi using **pip**:

Exercícios (usando venv)

1. Crie uma pasta chamada meu_projeto_venv e acesse essa pasta.
2. Crie um ambiente virtual com venv com o nome venv.
3. Ative o ambiente virtual.
4. Instale o pacote Flask.
5. Verifique se o Flask foi instalado corretamente.
6. Gere um arquivo requirements.txt com as dependências instaladas.
7. Atualize o pacote Flask para a versão mais recente.
8. Desinstale o pacote Flask.
9. Desative o ambiente virtual.

Exercícios (usando pipenv)

1. Crie uma nova pasta chamada `meu_projeto_pipenv` e acesse essa pasta.
2. Inicialize um novo ambiente virtual com pipenv e instale o pacote Flask.
3. Verifique se o ambiente foi criado e veja o conteúdo dos arquivos `Pipfile` e `Pipfile.lock`.
4. Instale o pacote `Flask-Login` como uma nova dependência.
5. Atualize o Flask para a versão mais recente com pipenv.
6. Liste todas as dependências do projeto gerando o gráfico de dependências.
7. Remova completamente o ambiente virtual criado por pipenv (use `pipenv --rm`).

Dúvidas



PROGRAMAÇÃO WEB II

Curso Técnico Integrado em Informática
Lucas Sampaio Leite

