PATHLIB E SHUTIL EM PYTHON Path

Caminhos e arquivos no Python — baseado na documentação oficial 3.14.6

Prof. Gustavo Franz (Science/Biology)
Python Developer in Progress
Building Educational Solutions with Python

GitHub: github.com/GustaFranz

Por que Path em vez de string?

String x Path

Aspecto	String	Path
`Juntar pastas`	'pasta' + '/' + 'arquivo'	pasta / 'arquivo'
`Portabilidade`	Barras \ ou / manual	Ajusta ao sistema
`Métodos úteis`	os.path separado	.exists(), .glob(), etc.
`Legibilidade`	Caminho longo confuso	Objeto com propriedades

Importar Path e operador /

Path é a classe principal para caminhos concretos. Ela representa um caminho real no disco da plataforma em que o código roda.

```
from pathlib import Path

pasta = Path('dados')
arquivo = pasta / 'simulado_7ano.csv'
print(arquivo)  # dados/simulado_7ano.csv
```

__file__ e .parent

__file__ guarda o caminho do script em execução. Com .parent, você sobe um nível na árvore de pastas — essencial para achar dados/ sem caminho fixo.

```
pasta_projeto = Path(__file__).resolve().parent
pasta_dados = pasta_projeto / 'dados'
print(pasta_projeto)
print(pasta_dados.exists())
```

Regra de ouro

- ✓ Nunca use C:/Users/... fixo no código.
- ✓ Use Path(__file__).parent para achar a pasta do exercício.

Árvore de pastas do projeto boletim

No projeto automacao-dashboard-boletim-escolar, os módulos ficam em src/ e acessam dados/ e saidas/ na raiz com .parent.parent.

Estrutura de pastas

```
automacao-dashboard-boletim-escolar/
    ■■■ src/
    ■■■   leitor_simulados.py
    ■■■   automacao_portal.py
    ■■■   consolidacao.py
    ■■■ dados/
    ■■■   simulados/
    ■■■   provas/
    ■■■ saidas/
    ■■■   boletins/
```

```
# Script em src/main.py
raiz = Path(__file__).resolve().parent.parent
pasta_simulados = raiz / 'dados' / 'simulados'
pasta_saidas = raiz / 'saidas' / 'boletins'
```

Fluxo:

src/main.py → .parent → .parent → raiz do projeto

Propriedades do caminho

```
caminho = Path('dados/simulado_7ano.csv')

caminho.name      # simulado_7ano.csv
caminho.stem      # simulado_7ano
caminho.suffix    # .csv
caminho.parent    # dados
caminho.parts     # ('dados', 'simulado_7ano.csv')
```

Referência rápida

Propriedade	Retorna	Exemplo
`.name`	Nome com extensão	notas.csv
`.stem`	Nome sem extensão	notas
`.suffix`	Extensão	.csv
`.parent`	Pasta pai	dados/
`.parts`	Tupla de partes	('/', 'etc', 'hosts')

exists(), is_file(), is_dir()

Fluxo:

Caminho → exists()? → is_file()? → Agir com segurança

```
arquivo = pasta_dados / 'simulado_7ano.csv'

if arquivo.exists():
    print('Arquivo encontrado')
if arquivo.is_file():
    print('É um arquivo')
if pasta_dados.is_dir():
    print('É uma pasta')
```

Padrão do projeto real

- ✓ Sempre verificar .exists() antes de ler ou mover.
- ✓ Evita FileNotFoundError e mensagens confusas.
- ✓ Usado em automacao_portal.py antes do shutil.move.

glob() e rglob() — buscar arquivos

glob() busca na pasta atual; rglob() busca recursivamente em subpastas. Ambos retornam um iterador de objetos Path.

Padrões comuns

Padrão	O que encontra
`'*.csv'`	CSV na pasta atual
`'**/*.csv'`	CSV em qualquer subpasta
`'simulado_*.csv'`	CSV com prefixo simulado_
`'*.xlsx'`	Planilhas Excel

```
pasta = Path('dados')
for csv in pasta.glob('*.csv'):
    print(csv)

for csv in pasta.rglob('**/*.csv'):
    print(csv.resolve())
```

mkdir(), touch(), unlink()

```
pasta_saida = raiz / 'saidas' / 'boletins'
pasta_saida.mkdir(parents=True, exist_ok=True)

arquivo = pasta_saida / 'aluno_01.html'
arquivo.touch()       # cria vazio se não existir
arquivo.unlink()      # apaga o arquivo
```

Método	Função	Parâmetro importante
`mkdir()`	Cria pasta	parents=True, exist_ok=True
`touch()`	Cria arquivo vazio	—
`unlink()`	Remove arquivo	Verificar exists() antes
`rename()`	Renomeia/move	Destino não pode existir*

parents=True cria pastas intermediárias. exist_ok=True evita erro se a pasta já existir.

read_text() e write_text()

Métodos convenientes para ler e gravar texto. Sempre use encoding='utf-8' para acentos e caracteres especiais.

```
conteudo = arquivo.read_text(encoding='utf-8')

html = '<html><body><h1>Boletim</h1></body></html>'
destino = pasta_saida / 'boletim_aluno.html'
destino.write_text(html, encoding='utf-8')
```

Exemplo do projeto boletim

- ✓ consolidacao.py grava CSV em saidas/
- ✓ boletins.py gera HTML por aluno com write_text.
- ✓ Sempre criar pasta de saída com mkdir antes.

resolve(), absolute(), relative_to()

```
p = Path('dados/../dados/simulado.csv')
p.resolve()      # caminho absoluto, sem ..
p.absolute()     # absoluto a partir do cwd

base = Path('/projeto')
arquivo = base / 'dados/notas.csv'
arquivo.relative_to(base)  # dados/notas.csv
```

Fluxo:

Path(__file__) → .resolve() → .parent → .parent → raiz

resolve() elimina .. e links simbólicos quando possível. Use para exibir caminhos completos ou comparar localizações.

Shutil — operações de alto nível

O módulo shutil complementa o pathlib: pathlib localiza caminhos; shutil copia, move e remove arquivos e pastas inteiras.

Qual função de cópia usar?

Função	Copia conteúdo	Copia metadados
`copyfile()`	Sim	Não
`copy()`	Sim	Permissões
`copy2()`	Sim	Permissões + datas
`copystat()`	Não	Só metadados

```
import shutil

shutil.copy('origem.csv', 'destino.csv')
shutil.copy2('origem.csv', 'backup.csv')
```

move(), copytree(), rmtree()

Fluxo:

Downloads/ → exists()? → unlink destino → shutil.move → dados/simulados/

```
def mover_csv_baixado(nome: str) -> None:
    origem = pasta_downloads / nome
    destino = pasta_simulados / nome
    if not origem.exists():
        print('Origem não encontrada')
        return
    if destino.exists():
        destino.unlink()
    shutil.move(origem, destino)
    print(f'Arquivo movido para: {destino}')
```

Funções extras úteis:

```
shutil.copytree('src_pasta', 'dst_pasta')
shutil.rmtree('pasta_temp')
shutil.disk_usage('/')  # total, usado, livre
shutil.which('python')  # caminho do executável
```

Exercício integrado

Combine pathlib + shutil: (1) glob() para achar CSV em downloads/, (2) mkdir em dados/simulados/, (3) move com shutil, (4) confirme com exists().

Prof. Gustavo Franz (Science/Biology)
Python Developer in Progress
Building Educational Solutions with Python

Professor de Ciencias e Biologia desde 2013 — atualmente estudando Programacao.

Eu crio estes resumos para organizar os assuntos e estudar de forma clara e objetiva. Estou compartilhando porque eles tambem podem ajudar outros desenvolvedores que estao comecando. Bons estudos — vamos aprender juntos!

GitHub: github.com/GustaFranz

Exercicios Python: github.com/GustaFranz/python_exercises
Para quem quiser ver a resolucao dos meus exercicios em Python.

EasyAnsi: github.com/GustaFranz/easyansi
Para quem quiser codificar personalizando com cores e estilos de forma mais facil, pratica e intuitiva.