Pathlib e Shutil

Caminhos e arquivos no Python

Para quem e este guia

- Quem quer parar de usar strings frageis como C:\Users\...\arquivo.csv.
- Estudantes do projeto de boletim escolar e automacao de dados.
- Quem precisa mover, copiar, criar pastas e ler arquivos com seguranca.

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
Metodos uteis	os.path separado	.exists(), .glob(), etc.
Legibilidade	Caminho longo confuso	Objeto com propriedades

Indice do guia

Este guia traduz a documentacao oficial para linguagem simples, com exemplos do projeto de boletim escolar.

Importar Path e operador /

Path e a classe principal para caminhos concretos. Ela representa um caminho real no disco da plataforma em que o codigo 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 execucao. Com .parent voce sobe um nivel na arvore 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 codigo.
Use Path(__file__).parent para achar a pasta do exercicio.

Arvore de pastas do projeto boletim

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

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')
```

Referencia rapida

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

exists(), is_file(), is_dir()

Fluxo:

Caminho -> exists()? -> is_file()? -> Agir com seguranca

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

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

Padrao 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. Retornam um iterador de objetos Path.

Padroes comuns

Padrao	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 nao existir
arquivo.unlink()      # apaga o arquivo
```

Metodo	Funcao	Parametro importante
mkdir()	Cria pasta	parents=True, exist_ok=True
touch()	Cria arquivo vazio	.
unlink()	Remove arquivo	Verificar exists() antes
rename()	Renomeia/move	Destino nao pode existir*

parents=True cria pastas intermediarias. exist_ok=True evita erro se a pasta ja existir.

read_text() e write_text()

Metodos 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 saida 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 simbolicos quando possivel. Use para exibir caminhos completos ou comparar localizacoes.

Shutil · operacoes de alto nivel

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

Qual funcao de copia usar?

Funcao	Copia conteudo	Copia metadados
copyfile()	Sim	Nao
copy()	Sim	Permissoes
copy2()	Sim	Permissoes + datas
copystat()	Nao	So 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 nao encontrada')
        return
    if destino.exists():
        destino.unlink()
    shutil.move(origem, destino)
    print(f'Arquivo movido para: {destino}')
```

Funcoes extras uteis:

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

Exercicio integrado

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