Manipulando arquivos e diretórios com PathLib

Live de Python # 199

Roteiro

1. Manipulação de arquivos

Uma introdução ao sistema de arquivos

2. PahtLib

O básico necessário

3. Resolvendo problemas

Usando a pathlib

4. Extras

PathLib3x, aioPathlib

picpay.me/dunossauro

apoia.se/livedepython

pix.dunossauro@gmail.com

Ajude o projeto <3

Acássio Anjos, Ademar Peixoto, A Earth, Alexandre Harano, Alexandre Souza, Alexandre Takahashi, Alexandre Villares, Alex Lima, Alynne Ferreira, Alysson Oliveira, Ana Carneiro, Ana Padovan, Andre Azevedo, André Rocha, Aquiles Coutinho, Arnaldo Turque, Ayrton Freeman, Bloquearsites Farewall, Bruno Barcellos, Bruno Freitas, Bruno Guizi, Bruno Oliveira, Bruno Ramos, Caio Nascimento, César Almeida, Christiano Morais, Clara Battesini, Cleber Santos, Daniel Haas, Danilo Segura, Dartz Dartz, David Kwast, Delton Porfiro, Dhyeives Rodovalho, Diego Guimarães, Dilenon Delfino, Donivaldo Sarzi, Douglas Bastos, Douglas Braga, Douglas Martins, Douglas Zickuhr, Emerson Rafael, Érico Andrei, Eugenio Mazzini, Euripedes Borges, Evandro Avellar, Fabiano Gomes, Fabio Barros, Fábio Barros, Fabio Castro, Fábio Thomaz, Felipe Rodrigues, Fernanda Prado, Flávio Meira, Flavkaze Flavkaze, Franklin Silva, Gabriel Barbosa, Gabriel Simonetto, Geandreson Costa, Guilherme Felitti, Guilherme Gall, Guilherme Ostrock, Gustavo Dettenborn, Heitor Fernandes, Henrique Junqueira, Igor Taconi, Ismael Ventura, Israel Gomes, Italo Silva, Jair Andrade, Janael Pinheiro, João Lugão, Johnny Tardin, Jonatas Leon, Jonatas Oliveira, Jônatas Silva, Jorge Plautz, Jose Mazolini, Juan Gutierrez, Juliana Machado, Julio Silva, Kaio Peixoto, Kaneson Alves, Leandro Botassio, Leandro Miranda, Leonardo Cruz, Leonardo Mello, Leonardo Nazareth, Lucas Adorno, Lucas Mello, Lucas Mendes, Lucas Oliveira, Lucas Polo, Lucas Praciano, Lucas Teixeira, Lucas Valino, Luciano Silva, Luciano Teixeira, Luiz Junior, Luiz Lima, Maiquel Leonel, Marcelino Pinheiro, Marcelo Matte, Márcio Martignoni, Marco Mello, Marcos Gomes, Marco Yamada, Maria Clara, Marina Passos, Mario Deus, Matheus Silva, Matheus Vian, Murilo Andrade, Murilo Cunha, Murilo Viana, Natan Cervinski, Nicolas Teodosio, Osvaldo Neto, Patricia Minamizawa, Patrick Brito, Paulo Tadei, Pedro Henrique, Pedro Pereira, Peterson Santos, Priscila Santos, Rafael Lopes, Rafael Romão, Ramayana Menezes, Reinaldo Silva, Renan Moura, Renato Veirich, Richard Nixon, Riverfount Riverfount, Rodrigo Ferreira, Rodrigo Freire, Rodrigo Junior, Rodrigo Vaccari, Rogério Sousa, Ronaldo Silva, Rui Jr, Samanta Cicilia, Sara Selis, Thiago Araujo, Thiago Borges, Thiago Bueno, Thiago Curvelo, Thiago Moraes, Tony Dias, Victor Wildner, Vinícius Bastos, Vinicius Figueiredo, Vítor Gomes, Vitor Luz, Vlademir Souza, Vladimir Lemos, Wellington Abreu, Wesley Mendes, William Alves, Willian Lopes, Wilson Neto, Yury Barros

Obrigado você

O sistema de arquivos

FS

Sistema de arquivos

Afinal como manipulamos arquivos e pastas no sistema?

Usamos o **Sistema de arquivos**

https://pt.wikipedia.org/wiki/Sistema_de_ficheiros

Sistema de arquivos

Afinal como manipulamos arquivos e pastas no sistema?

Usamos o **Sistema de arquivos**

"O sistema de arquivos é responsável por dar nomes, armazenar e recuperar os arquivos no seu sistema operacional"

https://pt.wikipedia.org/wiki/Sistema_de_ficheiros

Sistemas de arquivos

O que devemos considerar é que diferentes sistemas operacionais usam diferentes sistemas de arquivos.

- Windows
  - FAT
  - FAT32
  - NTFS
- Linux
  - EXT4
  - BtrFS
  - ZFS
- MacOS
  - APFS

Sistemas de arquivos

O que devemos considerar é que diferentes sistemas operacionais usam diferentes sistemas de arquivos.

- Windows
  - FAT
  - FAT32
  - NTFS
- Linux
  - EXT4
  - BtrFS
  - ZFS
- MacOS
  - APFS

MS rules
???

POSIX

Sistemas de arquivos

O que devemos considerar é que diferentes sistemas operacionais usam diferentes sistemas de arquivos.

- Windows
  - FAT
  - FAT32
  - NTFS
- Linux
  - EXT4
  - BtrFS
  - ZFS
- MacOS
  - APFS

MS rules
???

POSIX

Interface portável entre sistemas operacionais

Diferentes sistemas de arquivos

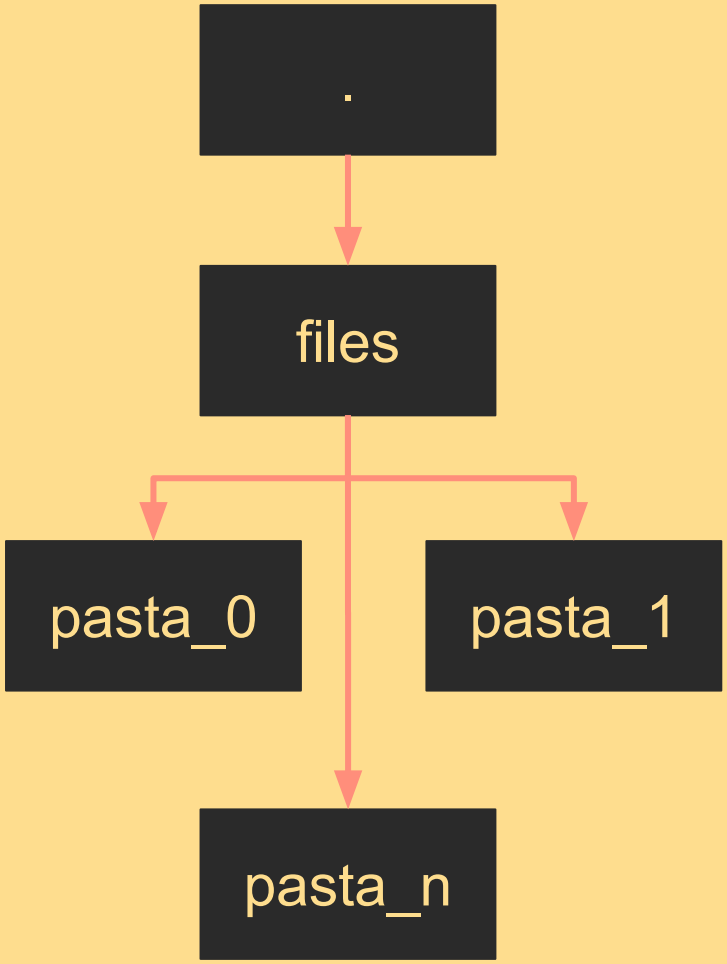
De forma totalmente superficial. Um problema inicial é

Os caminhos de diretórios/pastas/arquivos mudam de acordo com seu sistema operacional.

O mesmo código escrito no windows, não funciona no linux e no mac

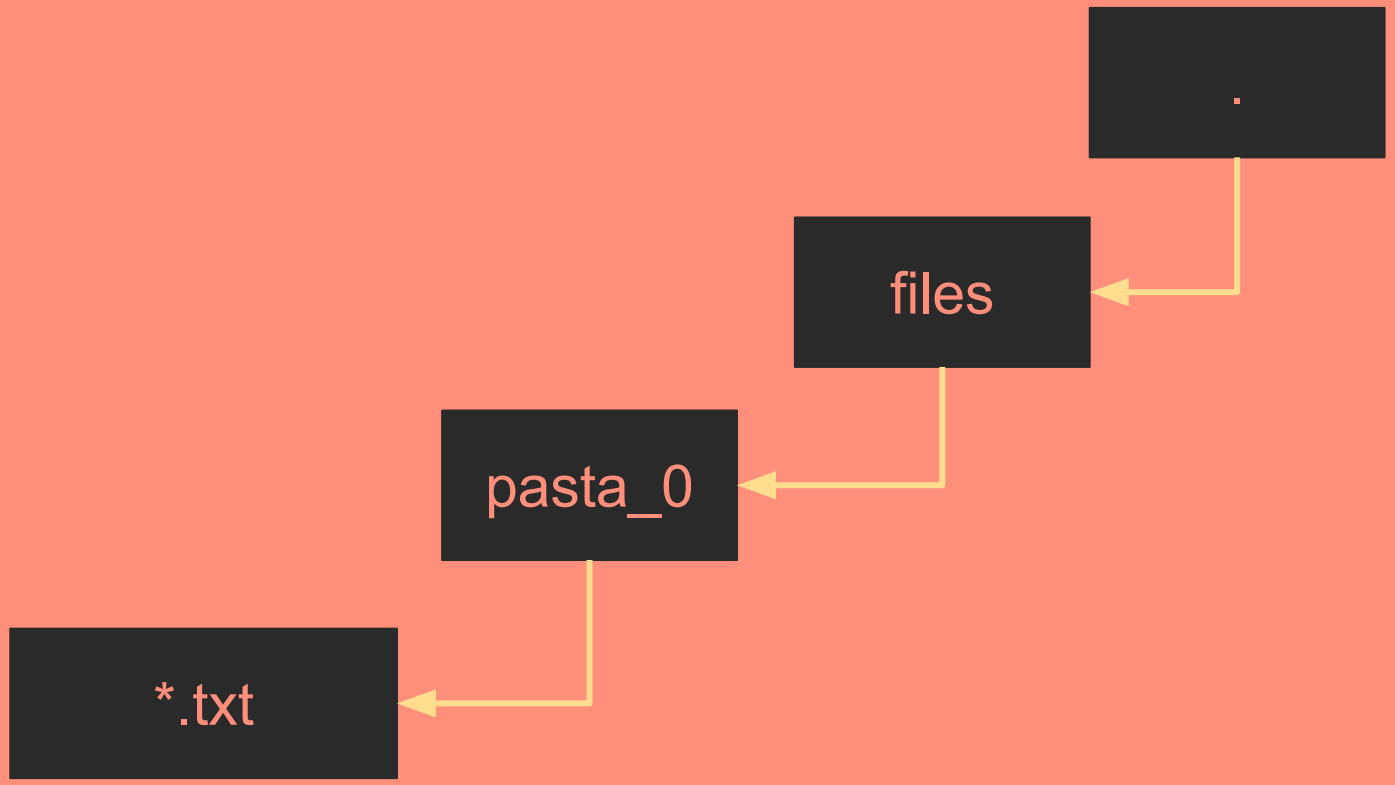
Um exemplo simples

```
py-3.10.4 babbage in ~/live_199
○ → exa -T -D .
.
└── files
    ├── pasta_0
    ├── pasta_1
    ├── pasta_2
    ├── pasta_3
    ├── pasta_4
    ├── pasta_5
    ├── pasta_6
    ├── pasta_7
    ├── pasta_8
    └── pasta_9
```



O desafio

Listar a **pasta_0**, que está dentro da pasta **files**, caso existam arquivos **.txt** dentro da **pasta_0**, printar o conteúdo dos arquivos.



O desafio

Listar a **pasta_0**, que está dentro da pasta **files**, caso existam arquivos **.txt** dentro da **pasta_0**, printar o conteúdo dos arquivos.

```
1   from glob import glob
2
3
4   for file in glob("files/pasta_0/*.txt"):
5       with open(file) as arquivo_txt:
6           print(arquivo_txt.read())
```

Listar a **pasta_0**, que está dentro da pasta **files**, caso existam arquivos **.txt** dentro da **pasta_0**, crie uma nova pasta chamada **desafio_2**, e copie os arquivos para essa pasta.

Desafio 2

O desafio 2

```python
import os
import shutil
from glob import glob
from os import path

if path.exists('desafio_2'):
    shutil.rmtree('desafio_2')

os.mkdir('desafio_2')


for file in glob('files/pasta_0/*.txt'):
    shutil.copy(file, 'desafio_2')
```

O desafio 2

```python
import os
import shutil
from glob import glob
from os import path

if path.exists('desafio_2'):
    shutil.rmtree('desafio_2')

os.mkdir('desafio_2')


for file in glob('files/pasta_0/*.txt'):
    shutil.copy(file, 'desafio_2')
```

API de baixo nível

Quando falamos de comunicação com o sistema operacional, e o FS, no nosso caso. O Python conta com uma gama de bibliotecas

os — Diversas interfaces de sistema operacional

Código-fonte: Lib/os.py

Este módulo fornece uma maneira simples de usar funcionalidades que são dependentes de sistema operacional. Se você quiser ler ou escrever um arquivo, veja `open()`; se o que quer é manipular estruturas de diretórios, veja o módulo `os.path`; e se quiser ler todas as linhas, de todos os arquivos, na linha de comando, veja o módulo `fileinput`. Para criar arquivos e diretórios temporários, veja o módulo `tempfile`; e, para manipulação em alto nível de arquivos e diretórios, veja o módulo `shutil`.

https://docs.python.org/pt-br/3/library/os.html

API de baixo nível

Quando falamos de comunicação com o sistema operacional, e o FS, no nosso caso. O Python conta com uma gama de bibliotecas

os — Diversas interfaces de sistema operacional

Código-fonte: Lib/os.py

Este módulo fornece uma maneira simples de usar funcionalidades que são dependentes de sistema operacional. Se você quiser ler ou escrever um arquivo veja `open()`; se o que quer é manipular estruturas de diretórios, veja o módulo `os.path`; e se quiser ler todas as linhas, de todos os arquivos, na linha de comando, veja o módulo `fileinput`. Para criar arquivos e diretórios temporários, veja o módulo `tempfile`; e, para manipulação em alto nível de arquivos e diretórios, veja o módulo `shutil`.

https://docs.python.org/pt-br/3/library/os.html

API de baixo nível

os — Diversas interfaces de sistema operacional

os.path — Manipulações comuns de nome nomes de caminhos

Código-fonte: Lib/posixpath.py (para POSIX) e Lib/ntpath.py (para Windows NT).

This module implements some useful functions on pathnames. To read or write files see `open()`, and for accessing the filesystem see the `os` module. The path parameters can be passed as strings, or bytes, or any object implementing the `os.PathLike` protocol.

Ao contrário de um shell Unix, Python não faz nenhuma expansão *automática* de caminho. Funções como `expanduser()` e `expandvars()` podem ser invocadas explicitamente quando uma aplicação deseja uma expansão de caminho no estilo do shell. (Veja também o módulo `glob`.)

https://docs.python.org/pt-br/3/library/os.path.html#module-os.path

API de baixo nível

os — Diversas interfaces de sistema operacional

os.path — Manipulações comuns de nome nomes de caminhos

Código-fonte: Lib/posixpath.py (para POSIX) e Lib/ntpath.py (para Windows NT).

This module implements some useful functions on pathnames. To read or write files see `open()`, and for accessing the filesystem see the `os` module. The path parameters can be passed as strings, or bytes, or any object implementing the `os.PathLike` protocol.

Ao contrário de um shell Unix, Python não faz nenhuma expansão *automática* de caminho. Funções como `expanduser()` e `expandvars()` podem ser invocadas explicitamente quando uma aplicação deseja uma expansão de caminho no estilo do shell. (Veja também o módulo `glob`.)

https://docs.python.org/pt-br/3/library/os.path.html#module-os.path

API de baixo nível

os — Diversas interfaces de sistema operacional

os.path — Manipulações comuns de nome nomes de caminhos

Código-fonte: Lib/posixpath.py (para POSIX) e Lib/ntpath.py (para Windows NT).

shutil — Operações de arquivo de alto nível

Código-fonte: Lib/shutil.py

O módulo `shutil` oferece várias operações de alto nível em arquivos e coleções de arquivos. Em particular, são fornecidas funções que possuem suporte a cópia e remoção de arquivos. Para operações em arquivos individuais, veja também o módulo `os`.

https://docs.python.org/pt-br/3/library/shutil.html

API de baixo nível

os — Diversas interfaces de sistema operacional

os.path — Manipulações comuns de nome nomes de

shutil — Operações de arquivo de alto nível

:mod:glob — Expansão de padrão de nome de arquivo no estilo Unix

Código-fonte: Lib/glob.py

O módulo `glob` encontra todos os nomes de caminho que correspondem a um padrão especificado de acordo com as regras usadas pelo shell Unix, embora os resultados sejam retornados em ordem arbitrária. Nenhuma expansão de til é feita, mas `*`, `?` e os intervalos de caracteres expressos com `[]` serão correspondidos corretamente. Isso é feito usando as funções `os.scandir()` e `fnmatch.fnmatch()` em conjunto, e não invocando realmente um subshell. Observe que, ao contrário de `fnmatch.fnmatch()`, `glob` trata nomes de arquivos que começam com um ponto (`.`) como casos especiais. (Para expansão de til e variável de shell, use `os.path.expanduser()` e `os.path.expandvars()`.)

https://docs.python.org/pt-br/3/library/glob.html#module-glob

Caminhos de mão única

O python, via um hack (os.altsep) consegue converter paths POSIX em paths windows.

```
1  import os
2
3  if os.name == 'nt':
4      filename = r'\\files\\pasta_0\\arquivo_1.txt'
5  else:
6    filename = 'files/pasta_0/arquivo_1.txt'
7
8  with open(filename) as file:
9      print(file.read())
```

https://docs.python.org/pt-br/3/library/os.html#os.altsep

Path lib

Uma introdução

O que é a Pathlib?

A Pathlib é uma alternativa a todas as API de baixo nível

- os
- os.path
- glob
- stat
- posixpath / ntpath / macpath (Finado Python 2)
- ...

* Não cobre **shutil,** pois já é uma AP de alto nível

A história da pathlib

- PEP-355 - 2006
  - Um objeto Path, que implementa todas as operações de os, os.path e shtuil comuns para arquivos
  - **REJEITADA**
- Criação da **pathlib** - Jan/2012
  - Biblioteca externa que implementa a PEP-355
  - Com alguns detalhes extras
- **PEP-428** - jun/2012 - Python 3.4
  - Inclusão da biblioteca externa ao biblioteca padrão
- **Python 3.6** - 2016
  - Interoperabilidade com os módulos antigos

A ideia da Pathlib

A ideia da Pathlib é fornecer uma API orientada a objetos.

Assim como datatime é um objeto e não manipulação de strings e números.

Além de não ter os problemas relacionados aos paths do windows, fornecer uma API que permite que você manipule paths do linux no windows, por exemplo.

Bom, chega de enrolação

```python
from pathlib import Path

path_atual = Path()
```

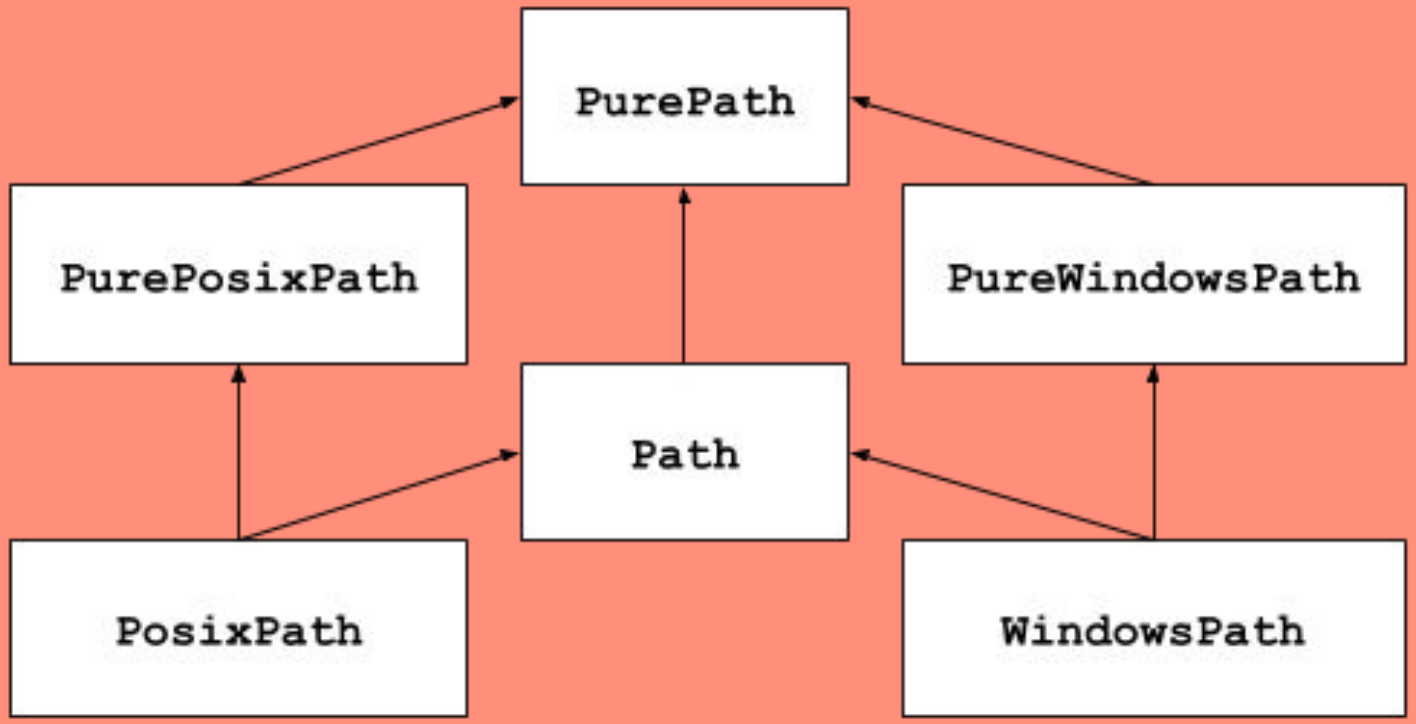
Alguns exemplos básicos - Path

```python
from pathlib import Path

path_atual = Path()
files = path_atual / 'files'

absoluto = files.absolute()
# PosixPath('/home/dunossauro/live_199/files')

files.name  # 'files'
absoluto.root  # '/'

absoluto.parent
# PosixPath('/home/dunossauro/live_199')

absoluto.parts
# ('/', 'home', 'dunossauro', 'live_199', 'files')
```

Alguns exemplos básicos - File

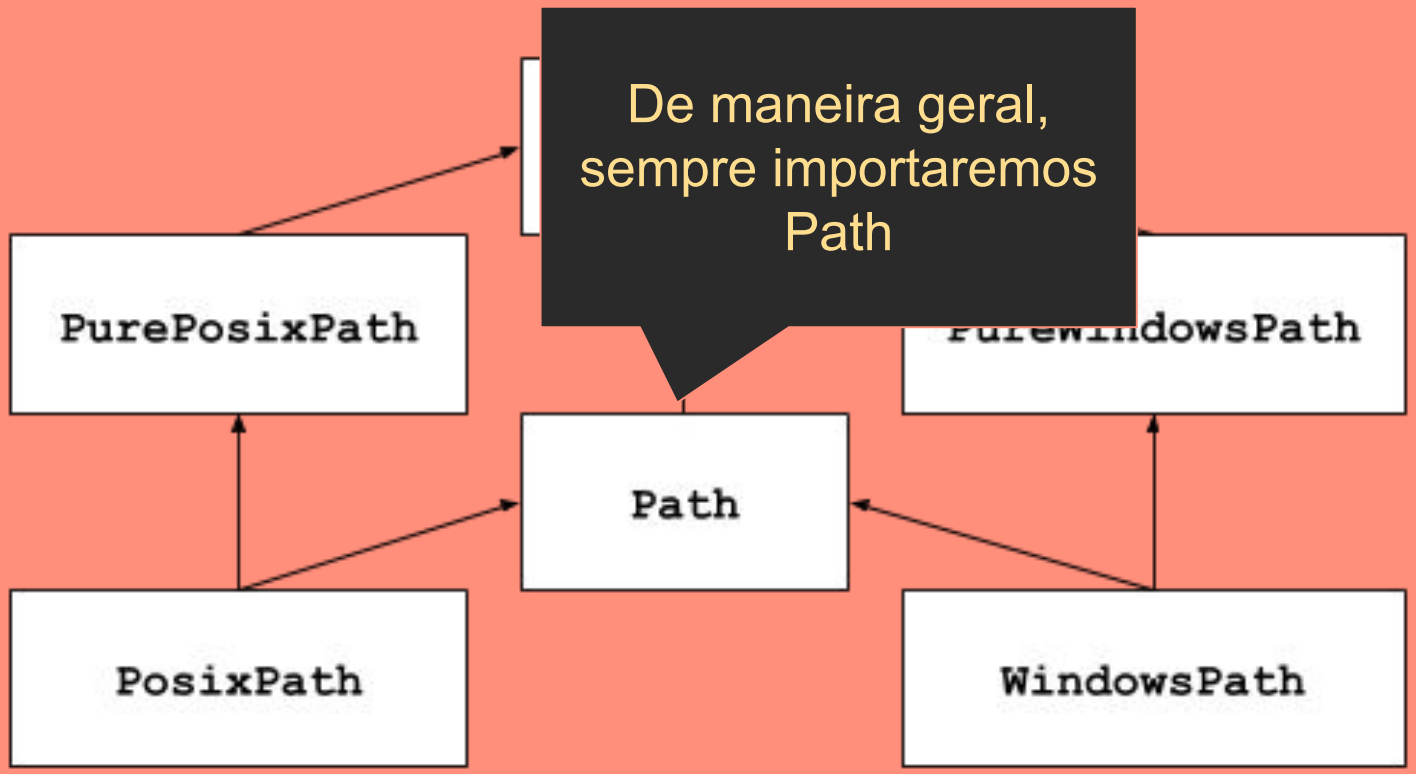
```python
path_atual = Path()
pasta_0 = path_atual / 'files' / 'pasta_0'
arquivo_0 = pasta_0 / 'arquivo_0.txt'

arquivo_0.exists()  # True
arquivo_0.is_file  # True

arquivo_0.suffix  # .txt
arquivo_0.stem  # arquivo_0

arquivo_0.read_text()
# 'files/pasta_0/arquivo_0.txt'
```

A estrutura da biblioteca



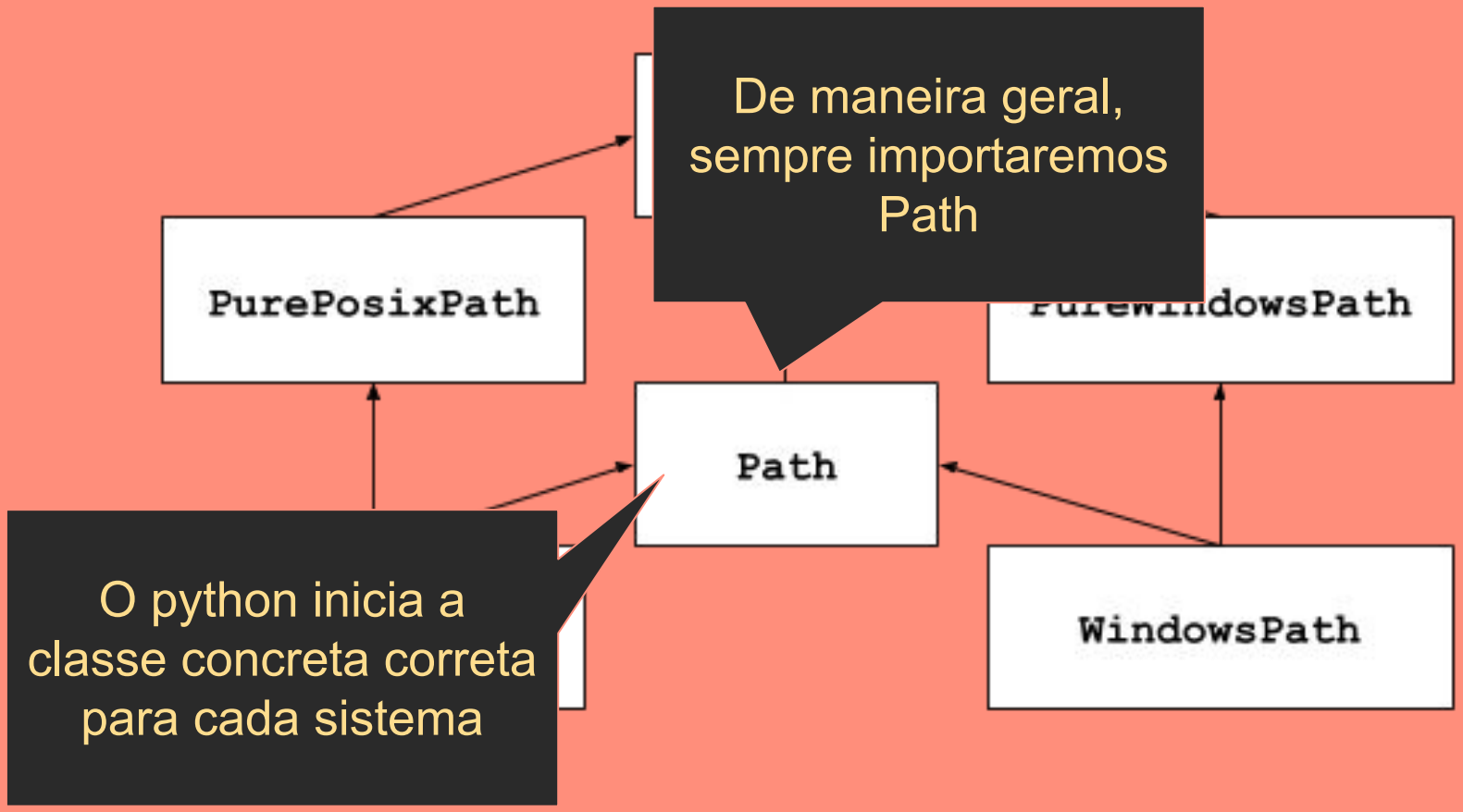
https://docs.python.org/pt-br/3/library/pathlib.html

A estrutura da biblioteca



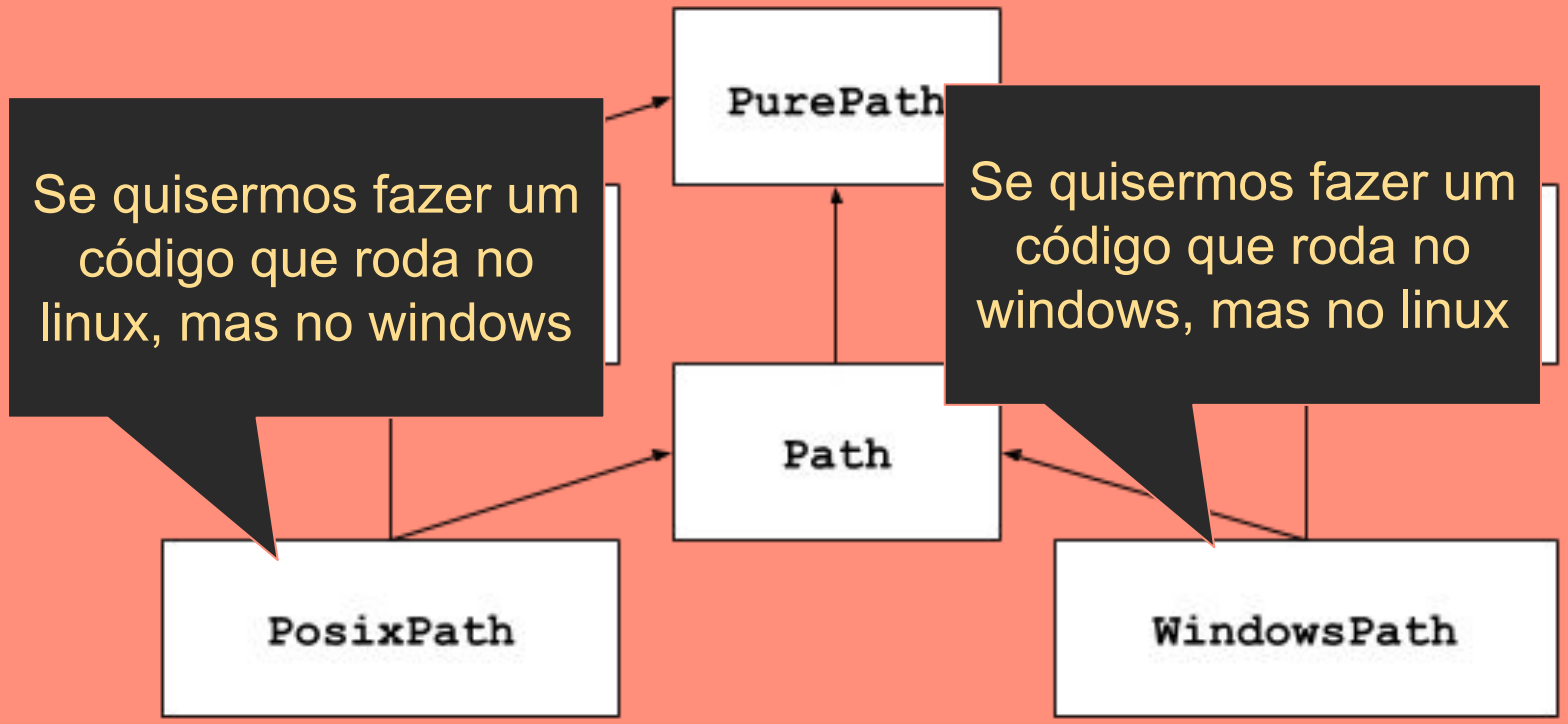
https://docs.python.org/pt-br/3/library/pathlib.html

A estrutura da biblioteca



https://docs.python.org/pt-br/3/library/pathlib.html

A estrutura da biblioteca



https://docs.python.org/pt-br/3/library/pathlib.html

Pathlib para manipular sistema de arquivos

- Arquivos
  - touch()
    - cria um novo arquivo ou altera seu timestamp
  - unlink()
    - remove um arquivo
- Paths
  - mkdir()
    - Cria uma pasta
  - rmdir()
    - remove uma pasta
- Tudo
  - rename
    - Renomeia o arquivo

Globing

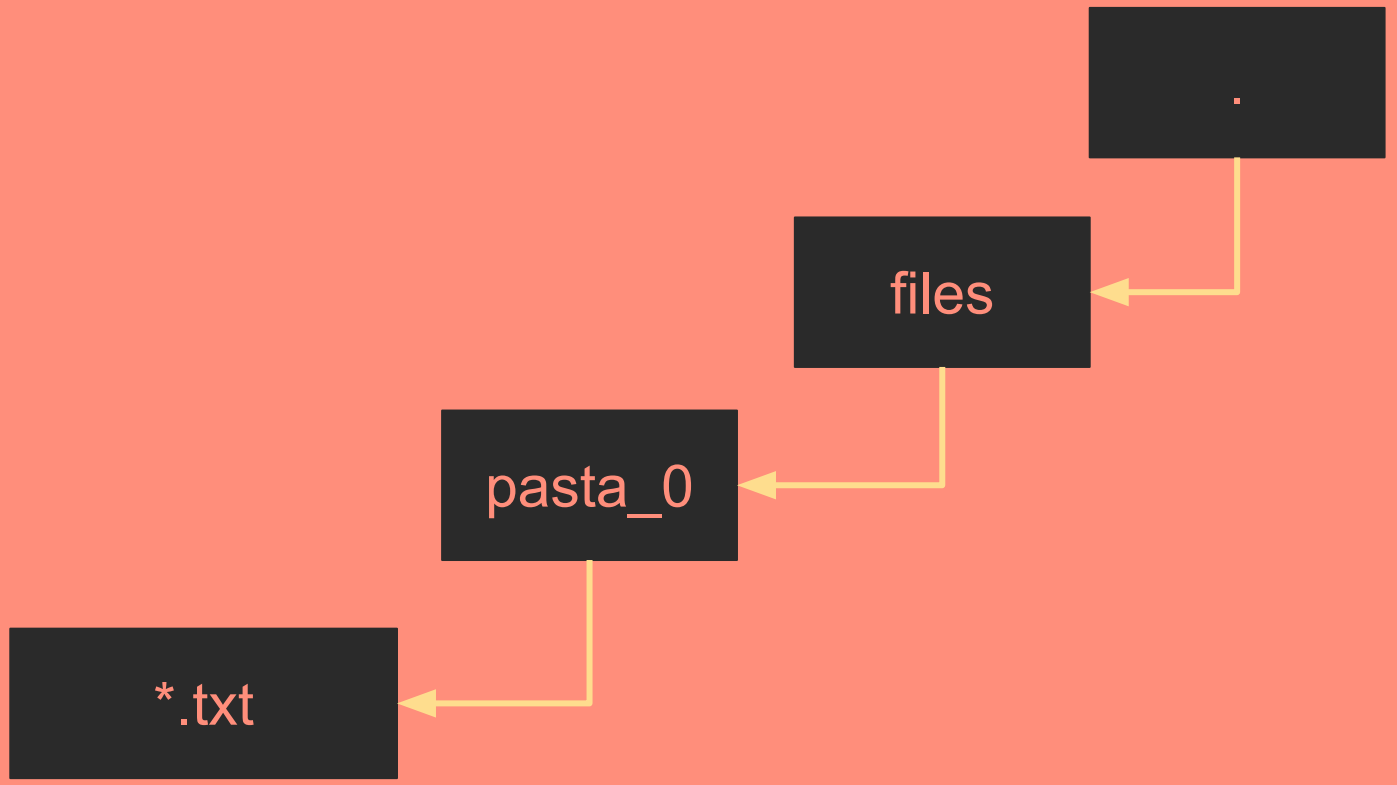
```
1   from pathlib import Path
2
3   p = Path('files')
4
5   p.glob(**/*.txt)
```

Que vamos resolver com Pathlib

Problemas

De volta ao problema 1

Listar a **pasta_0**, que está dentro da pasta **files**, caso existam arquivos **.txt** dentro da **pasta_0**, printar o conteúdo dos arquivos.



Solução com Pathlib

```python
from pathlib import Path

for file in Path('files/pasta_0').glob('*.txt'):
    print(file.read_text())
```

Comparação

```python
from pathlib import Path

for file in Path('files/pasta_0').glob('*.txt'):
    print(file.read_text())
```

```python
from glob import glob


for file in glob("files/pasta_0/*.txt"):
    with open(file) as arquivo_txt:
        print(arquivo_txt.read())
```

O problema 2

Listar a **pasta_0**, que está dentro da pasta **files**, caso existam arquivos **.txt** dentro da **pasta_0**, crie uma nova pasta chamada **desafio_2**, e copie os arquivos para essa pasta.

Solução com pathlib

```python
from pathlib import Path
from shutil import rmtree

output_path = Path('desafio_2')

if output_path.exists():
  # Sim, não tem rmtree na pathlib :(
  rmtree(output_path)

output_path.mkdir()

for file in Path('files/pasta_0').glob('*.txt'):
  new_file = output_path / file.name
  new_file.write_text(file.read_text())
```

Comparação

```python
from pathlib import Path
from shutil import rmtree

output_path = Path('desafio_2')

if output_path.exists():
  # Sim, não tem rmtree na pathlib :
  rmtree(output_path)

output_path.mkdir()

for file in Path('files/pasta_0').gl
  new_file = output_path / file.name
  new_file.write_text(file.read_text
```

```python
import os
import shutil
from glob import glob
from os import path

if path.exists('desafio_2'):
    shutil.rmtree('desafio_2')

os.mkdir('desafio_2')


for file in glob('files/pasta_0/*.txt'):
    shutil.copy(file, 'desafio_2')
```

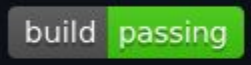
Extras

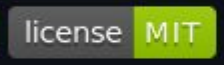
3x, async, ...

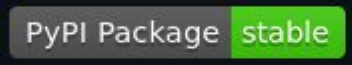
Sobre o shutil + pathlib

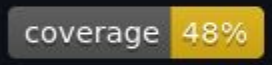
pathlib3x

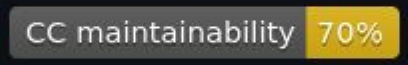
Version v1.3.9 as of 2020-10-09 see Changelog

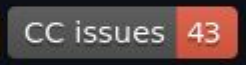


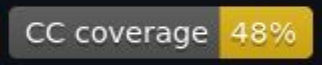












snyk

Backport of Python 3.10.0a0 pathlib for Python 3.6, 3.7, 3.8, 3.9 with a few tweaks to make it compatible.

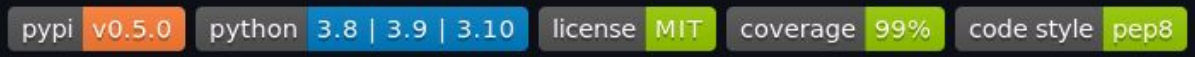
added wrappers to shutil copy, copy2, rmtree, copytree and other useful functions.

hmmmmmmm

```python
from pathlib3x import Path

output_path = Path('desafio_2')
output_path.rmtree(ignore_errors=True)
output_path.mkdir()

for file in Path('files/pasta_0').glob('*.txt'):
    new_file = output_path / file.name
    file.copy(new_file)
```

Asyncio + pathlib

aiopathlib: Pathlib support for asyncio



aiopathlib is written in Python, for handling local disk files in asyncio applications.

Base on aiofiles and just like pathlib, but use await.

```
with open('filename', 'w') as fp:
    fp.write('My file contents')

text = await aiopathlib.AsyncPath('filename').read_text()
print(text)
'My file contents'

content = await aiopathlib.AsyncPath(Path('filename')).read_bytes()
print(content)
b'My file contents'
```

Olha que lindo

```python
from asyncio import gather, run

async def uma_função_qualquer():
  await sleep(1)

async def lista_arquivos(path = AsyncPath('.'), pattern = '*'):
  path = AsyncPath(path)
  return [file async for file in path.glob(pattern)]

async def async_map():
  return await gather(lista_arquivos(), uma_função_qualquer())

run(async_map())
```

picpay.me/dunossauro

apoia.se/livedepython

pix.dunossauro@gmail.com

Ajude o projeto <3