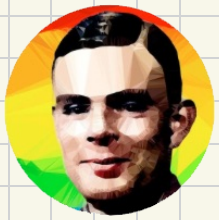
Una Ambiciosa Introducción a Python
Parte II



Pathlib en Python

En Python, la manipulación de rutas de archivos y directorios tradicionalmente se ha realizado con el módulo os. Sin embargo, la biblioteca pathlib, introducida en Python 3.4, proporciona una forma más intuitiva y orientada a objetos para trabajar con rutas y archivos.

1. Introducción a pathlib

Para usar pathlib, primero importamos la clase Path:
from pathlib import Path

La clase Path representa rutas de archivos o directorios en el sistema. Podemos usarla para realizar operaciones como listar archivos, verificar existencia, concatenar rutas y obtener propiedades de archivos.

2. Listar Directorios

Podemos obtener una lista de todos los directorios dentro de un directorio base con iterdir().

```
def list_directories(base_dir):
    print("# Listar directorios")
    path = Path(base_dir)
    dirs = [str(p) for p in path.iterdir() if p.is_dir()]
    print(f"Directorios: {dirs}")
```

- Path(base_dir): Convierte base_dir en un objeto Path.
- iterdir(): Itera sobre los elementos dentro de base_dir.
- is_dir(): Filtra solo los directorios.

3. Listar Archivos

Para listar solo los archivos en un directorio, usamos is_file().

```
def list_files(base_dir):
    print("# Listar archivos")
    path = Path(base_dir)
    files = [str(f) for f in path.iterdir() if f.is_file()]
    print(f"Archivos: {files}")
```

Lic. Franco Herrera

Una Ambiciosa Introducción a Python
Parte II

4. Buscar archivos con un patrón

Podemos buscar archivos que coincidan con un patrón usando glob().

```
def list_scripts(base_dir):
    print("# Listar scripts:")
    path = Path(base_dir)
    scripts = [str(s) for s in path.glob('**/*.py')]
    print(f"Scripts: {scripts}")
```

- glob('**/*.py'): Busca recursivamente (**/) todos los archivos .py en base_dir.

5. Concatenación de rutas

En pathlib, podemos usar el operador / para unir rutas de forma más clara que con os.path.join().

```
def concatenate_paths(base_dir):
    print("# Concatenar rutas con strings")
    path = Path(base_dir)
    print(f"path ({type(path)}): {path}")
    print(f"path / 'init.d' -> {path / 'init.d'}")
    p = path / 'init.d'
    print(f"p.parts -> {p.parts}")
```

salida

```
path (<class 'pathlib.PosixPath'>): /etc
path / 'init.d' -> /etc/init.d
p.parts -> ('/', 'etc', 'init.d')
```

6. Propiedades de rutas

Podemos verificar propiedades de archivos y directorios con pathlib

```
def show_properties():
    print(f"Existe .? {Path('.').exists()}")
    print(f"Existe temp? {Path('temp').exists()}")
    print(f"Es directorio .? {Path('.').is_dir()}")
    print(f"Es archivo .? {Path('.').is_file()}")
    print(f"Es absoluto .? {Path('.').is_absolute()}")
    print(f"Es relativo .? {Path('.').is_relative_to('.')}")
```

Una Ambiciosa Introducción a Python
Parte II

- exists(): Comprueba si la ruta existe.
- is_dir(), is_file(): Verifican si es directorio o archivo.
- is_absolute(): Indica si la ruta es absoluta.
- is_relative_to('.'): Comprueba si la ruta es relativa a otra

7. Obtener propiedades de Archivos

Podemos extraer información sobre archivos, como su extensión (suffix) y nombre (stem).

```
def show_file_elements(base_dir):
    p = Path(base_dir)
    print(f"p.suffix -> {p.suffix}")
    print(f"p.suffixes -> {p.suffixes}")
    print(f"p.stem -> {p.stem}")
```

Ejemplo con un archivo compressed.tar.bz2:
p.suffix -> .bz2
p.suffixes -> ['.tar', '.bz2']
p.stem -> compressed.tar

8. Gestión de permisos

Podemos verificar los permisos de un archivo usando el método .stat().st_mode

```
import stat

def check_permissions(file_path):
    path = Path(file_path)
    if not path.exists():
        print(f"El archivo {file_path} no existe.")
        return

    mode = path.stat().st_mode
    print(f"Permisos de {file_path}:")
    print(f"Lectura: {'Sí' if bool(mode & stat.S_IRUSR) else 'No'}")
    print(f"Escritura: {'Sí' if bool(mode & stat.S_IWUSR) else 'No'}")
    print(f"Ejecución: {'Sí' if bool(mode & stat.S_IXUSR) else 'No'}")
```

path.stat().st_mode obtiene los permisos del archivo.
stat.S_IRUSR, stat.S_IWUSR, stat.S_IXUSR permiten verificar lectura, escritura y ejecución.

Una Ambiciosa Introducción a Python
Parte II

Ejemplo:

check_permissions("script.py")

Permisos de script.py:
Lectura: Sí
Escritura: Sí
Ejecución: No

9. Cambiar permisos de un archivo

Podemos modificar los permisos de un archivo con chmod()

```
def change_permissions(file_path, read=True, write=True, execute=False):
    path = Path(file_path)
    if not path.exists():
        print(f"El archivo {file_path} no existe.")
        return

    mode = 0
    if read:
        mode |= stat.S_IRUSR
    if write:
        mode |= stat.S_IWUSR
    if execute:
        mode |= stat.S_IXUSR

    path.chmod(mode)
    print(f"Permisos cambiados para {file_path}.")
```

Ejemplo

change_permissions("script.py", read=True, write=False, execute=True)

Este comando otorga permisos de lectura y ejecución, pero elimina el permiso de escritura.

Una Ambiciosa Introducción a Python
Parte II

pathlib es un API moderna y más legible a os.path, ofreciendo:

- **Código más limpio**: Uso de objetos en lugar de cadenas.
- **Operaciones más intuitivas**: Concatenación con /, iteración directa sobre directorios.
- **Compatibilidad con Windows y Unix**: Path maneja automáticamente las diferencias de ruta (\ vs /).

Usar pathlib hace que el manejo de archivos y directorios sea más elegante y **Pythonic.**

Lic. Franco Herrera