GABRIELE ROMANATO

Ottenere informazioni sui file in Python

Ottenere informazioni su un file in Python è un'operazione essenziale quando si lavora con il file system, poiché permette di recuperare dettagli quali dimensioni, tipo, permessi e timestamp di modifica. Python offre diverse librerie e funzioni per eseguire queste operazioni, in particolare attraverso i moduli integrati `os`, `os.path`, `stat` e `pathlib`. In questo articolo, esploreremo i modi più comuni per ottenere informazioni su un file.

Il modulo `os` è uno strumento potente per interagire con il sistema operativo, e il sottopacchetto `os.path` contiene funzioni utili per gestire i file e le directory.

Per verificare l'esistenza di un file, è possibile usare la funzione `os.path.exists()`:

```
import os

file_path = 'file.txt'
if os.path.exists(file_path):
    print("Il file esiste.")
else:
    print("Il file non esiste.")
```

Per conoscere le dimensioni di un file, si può usare `os.path.getsize()`:

```
file_size = os.path.getsize(file_path)
print(f"La dimensione del file è: {file_size} byte")
```

Il modulo `os` permette di ottenere informazioni temporali su un file, come la data di creazione, di modifica e di accesso.

```
import time

mod_time = os.path.getmtime(file_path)
print("Ultima modifica:", time.ctime(mod_time))

access_time = os.path.getatime(file_path)
print("Ultimo accesso:", time.ctime(access_time))
```

```
create_time = os.path.getctime(file_path)
print("Data di creazione:", time.ctime(create_time))
```

Il modulo `stat` fornisce una maggiore flessibilità nella gestione delle informazioni sui file. Usando `os.stat()`, è possibile ottenere un oggetto contenente varie informazioni dettagliate.

```
import os
import stat

file_info = os.stat(file_path)
print("Dimensione del file:", file_info.st_size, "byte")
print("Ultima modifica:", time.ctime(file_info.st_mtime))
print("Ultimo accesso:", time.ctime(file_info.st_atime))
print("Data di creazione:", time.ctime(file_info.st_ctime))
```

Oltre alle dimensioni e ai timestamp, `os.stat()` restituisce anche informazioni sui permessi e altre proprietà del file.

Per controllare i permessi di lettura, scrittura ed esecuzione, si possono usare le costanti definite nel modulo `stat`.

```
if bool(file_info.st_mode & stat.S_IRUSR):
    print("Il file è leggibile.")
if bool(file_info.st_mode & stat.S_IWUSR):
    print("Il file è scrivibile.")
if bool(file_info.st_mode & stat.S_IXUSR):
    print("Il file è eseguibile.")
```

A partire da Python 3.4, il modulo `pathlib` è diventato un’alternativa potente e più intuitiva per lavorare con i percorsi dei file. Questo modulo semplifica molte operazioni che sarebbero più verbose usando `os` o `stat`.

Per ottenere informazioni su un file, si inizia creando un oggetto `Path` per il file desiderato.

```
from pathlib import Path

file_path = Path('file.txt')
```

Con `pathlib`, verificare l'esistenza di un file, controllare se è un file o una directory, e altre operazioni simili, diventano più semplici e leggibili:

```
if file_path.exists():
    print("Il file esiste.")
    if file_path.is_file():
        print("È un file.")
    elif file_path.is_dir():
        print("È una directory.")
```

Anche recuperare le dimensioni del file e i vari timestamp è semplice:

```
print("Dimensione del file:", file_path.stat().st_size, "byte")
print("Ultima modifica:", file_path.stat().st_mtime)
print("Ultimo accesso:", file_path.stat().st_atime)
print("Data di creazione:", file_path.stat().st_ctime)
```

`pathlib` permette inoltre di ottenere timestamp in formato `datetime`, rendendo il tutto più semplice e leggibile:

```
from datetime import datetime

mod_time = datetime.fromtimestamp(file_path.stat().st_mtime)
print("Ultima modifica:", mod_time)
```

Anche se `pathlib` non ha un supporto diretto per controllare i permessi, si può ottenere lo stesso risultato usando `file_path.stat().st_mode` e `stat`:

```
import stat

if file_path.stat().st_mode & stat.S_IRUSR:
    print("Il file è leggibile.")
if file_path.stat().st_mode & stat.S_IWUSR:
    print("Il file è scrivibile.")
if file_path.stat().st_mode & stat.S_IXUSR:
    print("Il file è eseguibile.")
```

Conclusione

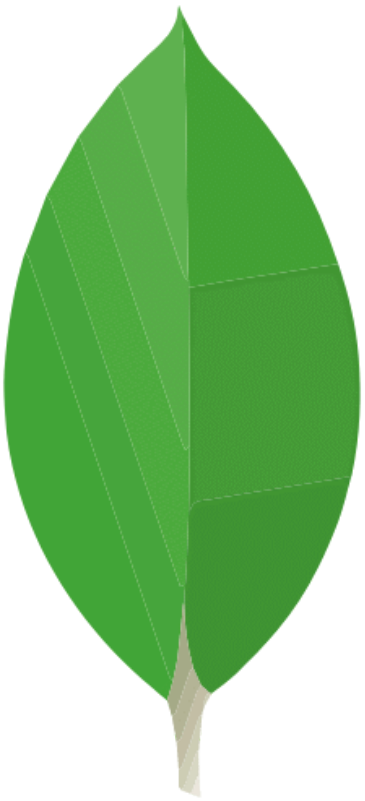
In Python, il recupero delle informazioni sui file può essere effettuato utilizzando `os`, `stat` o `pathlib`, ognuno dei quali ha i propri vantaggi. Il modulo `os` è ampiamente compatibile e standard, mentre `stat` fornisce una visione approfondita sui permessi e altre proprietà avanzate. Tuttavia, per la maggior parte delle operazioni moderne, `pathlib` è preferito per la sua leggibilità e semplicità.

Applicazioni Correlate

- ### Python Placeholder Image

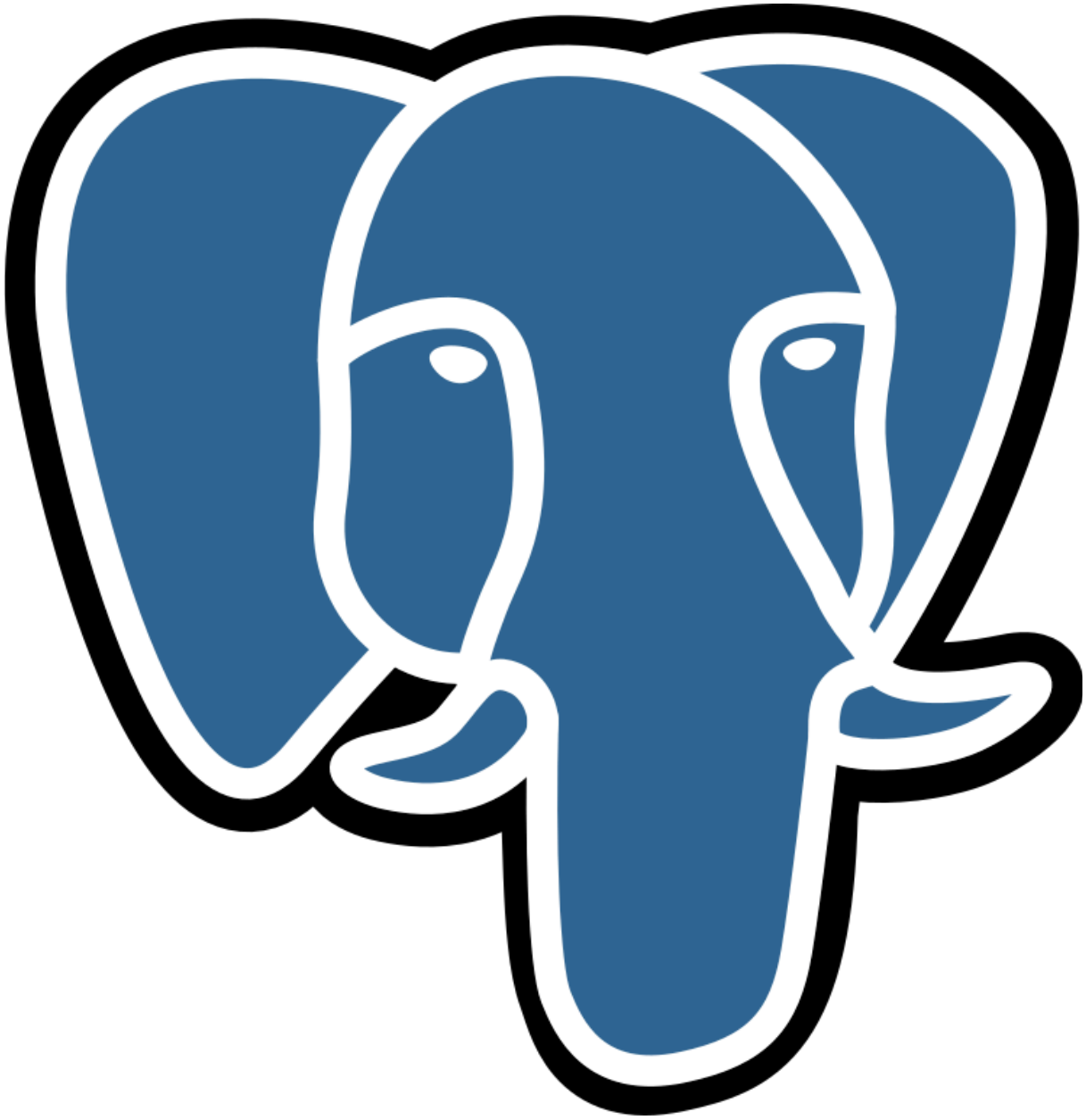
  Applicazione sviluppata in Python con Flask per la creazione di immagini segnaposto.

DockerDocker ComposePythonFlaskJavaScript



Python MongoDB App

Applicazione basata su MongoDB e sviluppata in Python e Flask.
DockerDocker ComposePythonFlaskMongoDB



Python PostgreSQL App

Applicazione basata su PostgreSQL con Python e Flask.
DockerDocker ComposePythonFlask