کار با سیستم عامل در پایتون: فراتر از مرزها

کتابخانه os در پایتون، ابزاری قدرتمند برای تعامل با سیستم عامل است. این کتابخانه به شما امکان می‌دهد تا بدون توجه به سیستم عامل مورد استفاده، کارهای سیستمی را انجام دهید و برنامه‌های قابل حمل بنویسید.

مدیریت مسیرها: پیمایش در دنیای فایل‌ها

- `os.getcwd()` : دریافت مسیر فعلی دایرکتوری.

```
import os

current_path = os.getcwd()
print(current_path)  # Output: /home/user/my_project
```

- `os.chdir(path)` : تغییر مسیر فعلی دایرکتوری.

```
os.chdir('/home/user/documents')
```

- `os.listdir(path)` : دریافت لیست فایل‌ها و دایرکتوری‌های موجود در یک مسیر.

```
files = os.listdir('.')  # Lists files in the current directory
print(files)
```

ایجاد و حذف فایل‌ها و دایرکتوری‌ها: ساختاردهی به داده‌ها

- `os.mkdir(path)` : ایجاد یک دایرکتوری جدید.

```
os.mkdir('new_folder')
```

- `os.rmdir(path)` : حذف یک دایرکتوری خالی.

```
os.rmdir('empty_folder')
```

- `os.remove(path)` : حذف یک فایل.

```
os.remove('file.txt')
```

- `os.rename(src, dst)` : تغییر نام یا انتقال یک فایل یا دایرکتوری.

```
os.rename('old_name.txt', 'new_name.txt')
os.rename('file.txt', 'folder/file.txt')
```

اجرای دستورات سیستمی: تعامل مستقیم با سیستم عامل

- `os.system(command)` : اجرای یک دستور سیستمی.

```
os.system('ls -l')  # Lists files in Linux/macOS
os.system('dir')    # Lists files in Windows
```

پیمایش در ساختار دایرکتوری: جستجوی عمیق

- `os.walk(top)` : پیمایش در یک دایرکتوری و زیردایرکتوری‌های آن.

```
for root, dirs, files in os.walk('.'):
    print(f"Directory: {root}")
    for file in files:
        print(f"  File: {file}")
```

کار با مسیرها: ابزارهای کمکی

- `os.path.exists(path)` : بررسی وجود یک فایل یا دایرکتوری.

```
if os.path.exists('file.txt'):
    print("File exists!")
```

- `os.path.isfile(path)` : بررسی اینکه آیا یک مسیر به یک فایل اشاره می‌کند.

```
if os.path.isfile('file.txt'):
    print("This is a file.")
```

- `os.path.isdir(path)` : بررسی اینکه آیا یک مسیر به یک دایرکتوری اشاره می‌کند.

```
if os.path.isdir('folder'):
    print("This is a directory.")
```

- `os.path.basename(path)` : دریافت نام فایل یا دایرکتوری از یک مسیر.

```
filename = os.path.basename('/path/to/file.txt')
print(filename)  # Output: file.txt
```

- `os.path.join(path1, path2, ...)` : ایجاد یک مسیر با استفاده از اجزای مختلف.

```
new_path = os.path.join('/home/user', 'documents', 'file.txt')
print(new_path)  # Output: /home/user/documents/file.txt
```

- `os.path.commonprefix(list)` : دریافت پیشوند مشترک بین چند مسیر.

```
common = os.path.commonprefix(['/home/user/file1.txt', '/home/user/file2.
print(common)  # Output: /home/user/
```

نکات کلیدی

- از `os.sep` برای جداکننده مسیر مناسب با سیستم عامل استفاده کنید.

- برای مدیریت خطاها، از بلوک‌های `try...except` استفاده کنید.
- از `os.path.join()` برای ساخت مسیرها به صورت قابل حمل استفاده کنید.