بسته‌بندی فایل‌ها

بسته‌بندی فایل‌ها

در طراحی و مدیریت پروژه‌های نرم‌افزاری، سازمان‌دهی صحیح فایل‌ها از اهمیت بالایی برخوردار است. اگر تمام بخش‌های یک پروژه را در یک فایل واحد ذخیره کنیم، پس از مدتی این فایل بیش‌ازحد حجیم و پیچیده شده و خوانایی و نگه‌داری آن دشوار خواهد شد. اما با تفکیک اجزای مختلف پروژه در قالب فایل‌ها و پوشه‌های مجزا، ساختار کد منظم‌تر شده و توسعه و خطایابی آن ساده‌تر می‌شود.

ماژول (Module)

ماژول‌ها یکی از ابزارهای کلیدی در مدیریت پروژه‌های پایتونی هستند. به‌طور کلی، هر فایل پایتونی که شامل توابع، متغیرها و کلاس‌های مختلف باشد، یک ماژول محسوب می‌شود. برای مثال، اگر دو فایل **test.py** و **main.py** داشته باشیم و در فایل **main.py** از قابلیت‌های **test.py** استفاده کنیم، درواقع **test.py** به‌عنوان یک ماژول عمل می‌کند.

دستور `import`

با استفاده از `import` می‌توان یک ماژول را در فایل دیگر بارگذاری کرد و به توابع و متغیرهای آن دسترسی یافت. برای مثال، فرض کنید که فایل **test.py** شامل کدهای زیر است:

```
def show_message():
    print("Hello from test.py!")

number = 42
```

اکنون در **main.py** می‌توان این ماژول را وارد کرده و از اجزای آن استفاده کرد:

```
import test
```

```
test.show_message()
print(test.number)
```

خروجی:

```
Hello from test.py!
42
```

دسترسی به ماژول‌های موجود در پوشه‌های مختلف

اگر فایل ماژول در مسیری متفاوت از فایل اصلی قرار داشته باشد، می‌توان با استفاده از مسیر نسبی آن را وارد کرد. به‌عنوان مثال، اگر پروژه ساختاری مانند زیر داشته باشد:

```
.
├── controller.py
├── main
│   ├── main.py
│   └── test.py
└── model.py
```

و بخواهیم در **controller.py** به فایل **test.py** درون پوشه **main** دسترسی داشته باشیم، باید از نام پوشه به‌عنوان بخشی از ماژول استفاده کنیم:

```
import main.test

main.test.show_message()
```

استفاده از `from` برای وارد کردن ویژگی‌های خاص

به‌جای وارد کردن کل ماژول، می‌توان تنها برخی از ویژگی‌های آن را به‌طور مستقیم بارگذاری کرد.

```
from test import show_message, number


```

```
show_message()
print(number)
```

در این روش نیازی به استفاده از `test.` برای دسترسی به ویژگی‌ها نیست. همچنین، برای وارد کردن تمام ویژگی‌های ماژول می‌توان از `*` استفاده کرد:

```
from test import *

show_message()
print(number)
```

تغییر نام ویژگی‌های واردشده با `as`

با استفاده از `as` می‌توان نام ماژول یا ویژگی‌های واردشده را تغییر داد:

```
from math import pi as p, sin as cosine

print(cosine(p / 2))
```

خروجی:

```
1.0
```

این روش به‌خصوص برای کوتاه کردن نام‌ها یا جلوگیری از تداخل نام‌ها در پروژه‌های بزرگ مفید است.