

এতদিন পাইথন ইন্টারপ্রেটারে কোডের ওপর আলোচনা করা হয়েছে, কিন্তু এরপরও ইন্টারপ্রেটার বন্ধ করলেই কোডগুলো মুছে যেত। তাই বড় প্রোগ্রাম লিখতে চাইলে ভালো হয় কোনো টেক্সট এডিটর দিয়ে ইন্টারপ্রেটারের জন্য ইনপুট লিখে ফাইলে সেভ করে তা ইনপুট করা। একে বলে 'স্ক্রিপ্ট' তৈরি করা। বড় প্রোগ্রামের ক্ষেত্রে রক্ষণাবেক্ষণের সুবিধার জন্য কোড ছোট ছোট অংশে ভাগ করে আলাদা ফাইলে সেভ করে রাখলে ভালো হয়। আবার একই কাজের জন্য আলাদা ফাইলে

এতে কস্মোলে নতুন কিছু দেখতে পাবেন না। কারণ, এখানে শুধু মডিউলটি ইম্পোর্ট করা হয়েছে, এর ফাংশনগুলো ব্যবহার করা হয়নি। এখানে মডিউল নাম ব্যবহার করে ফাংশনগুলো চালাতে পারেন।

```
>>> fibo.fib(500)
1 1 2 3 5 8 13 21 34 55 89
144 233 377
>>> fibo.fib2(500)
[1, 1, 2, 3, 5, 8, 13, 21, 34,
55, 89, 144, 233, 377]
>>> fibo.__name__
'fibo'
```

যদি প্রোগ্রামে একটি ফাংশন বারবার ব্যবহার করতে চান, তাহলে



ফাংশনটি কপি না করে একটি ফাইলে লিখে সেখান থেকেই দরকার অনুযায়ী কল করা যায়। এসব কাজের জন্য কোনো ফাংশন ফাইলে লিখে তা স্ক্রিপ্ট হিসেবে পাইথনে ব্যবহার করা যায়। এই ফাইলগুলোকে বলা হয় মডিউল। এক মডিউলের ফাংশন অন্য মডিউল থেকে ব্যবহার করা যায়। মডিউলে ফাংশন লিখে তা ফাইলে সেভ করার সময় ফাইল নামের শেষে .py যোগ করতে হবে। যেমন- একটি ফাইল নিন, যার নাম fibo.py এবং ফাইলটি খুলে নিচের কোডটি পেস্ট করে দিন।

```
def fib(n):
    # write Fibonacci series
    up to n
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()
def fib2(n):
    # return Fibonacci series
    up to n
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
এবার ফাইলটি সেভ করে রাখুন। এবার পাইথনের ইন্টারপ্রেটারে ঢুকে কমান্ড দিন
>>> import fibo
```

বারবার এত বড় নাম না লিখে লোকাল নাম দিয়ে রাখতে পারেন।

```
>>> f = fibo.fib
>>> f(500)
1 1 2 3 5 8 13 21 34 55 89
144 233 377
একটি মডিউলে অনেক ফাংশন থাকতে পারে। যদি সেখান থেকে নির্দিষ্ট কোনো ফাংশন নিতে, চাইলে এভাবে ইম্পোর্ট করতে পারেন-
>>> from fibo import fib
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89
144 233 377
```

আবার যদি পুরো মডিউলের সব ফাংশন নিতে চান ও শুধু ফাংশন নাম দিয়েই ফাংশন কল করতে চাইলে এভাবেও কল করতে পারেন-

```
>>> from fibo import *
আপনি চাইলে মডিউলকে স্ক্রিপ্ট হিসেবেও রান করতে পারেন। এর জন্য fibo.py ফাইলটি এডিট করতে হবে। ফাইলের শেষে নিচের লাইনগুলো যোগ করে সেভ করে রাখতে হবে।
```

```
if __name__ ==
"__main__":
    import sys
    fib(int(sys.argv[1]))
এখন কমান্ড প্রম্পট থেকে ফাইলটি রান করতে পারেন।
C:\Users\user\Desktop
>python fibo.py 200
1 1 2 3 5 8 13 21 34 55 89
144
```

কিন্তু fibo.py মডিউলটি ইম্পোর্ট করলে কিছু প্রিন্ট করবে না। কারণ, fibo.py যতক্ষণ পর্যন্ত মেইন ফাইল হিসেবে রান না করা হচ্ছে, ততক্ষণ পর্যন্ত কিছুই করবে না।

সফটওয়্যারকে দ্রুত রান করানোর জন্য পাইথন বারবার মডিউলগুলোকে কম্পাইল করে না, বরং একবারই কম্পাইল করে pycache নামের ডিরেক্টরিতে রেখে দেয়। পাইথন চেক করে দেখে সোর্স কবে শেষবার এডিট করা হয়েছে, যদি নতুন করে কিছু করা হয় তাহলে সে আবার মডিউলকে কম্পাইল করে। এটা পাইথনের অটোমেটিক প্রসেস। ফলে সব সময় উন্নত ভার্সনটি পাওয়া যায়। তবে কমান্ড প্রম্পট থেকে রান করা হলে বারবার মডিউলটি কম্পাইল হয়।

পাইথনের একটি স্ট্যান্ডার্ড মডিউল লাইব্রেরি আছে। কিছু মডিউল ইন্টারপ্রেটারের সাথে বিল্টইন হিসেবে থাকে, যেগুলো ইন্টারপ্রেটারকে অপারেটিং সিস্টেমের সাথে যোগাযোগের ব্যবস্থা করে দেয়। এরা পাইথনের কোর মডিউলের সাথে সম্পৃক্ত নয়। একটি মডিউলের উদাহরণ নিম্নরূপ-

```
>>> import sys
>>> sys.path
['', 'C:\\WINDOWS\\SYS-
TEM32\\python34.zip',
'C:\\Python34\\DLLs',
'C:\\Python34\\lib',
'C:\\Python34',
'C:\\Python34\\lib\\site-pack-
ages']
```

বিল্টইন ফাংশন dir() ব্যবহার করে মডিউলের ভেতরে কী কী আছে দেখতে পারি। যদি কোনো প্যারামিটার ছাড়াই dir()-কে কল করা স্ক্রিপ্টের সবগুলো ব্যবহৃত নামকেই লিস্ট হিসেবে দেখাবে।

```
>>> a = [1,2,3,4,5]
>>> import fibo
>>> fib = fibo.fib
>>> dir()
['_builtins_',
'_name_', 'a', 'fib', 'fibo']
>>> dir(fibo)
['_name_', 'fib', 'fib2']
ফাংশন dir() ব্যবহার করে বিল্টইন ফাংশন বা ভ্যারিয়েবল দেখা যায় না। তাই আমাদের builtins মডিউল ব্যবহার করতে হবে-
>>> import builtins
```

```
>>> dir(builtins)
['ArithmeticError', ...,
'vars', 'zip']
```

পাইথনের মডিউল স্ট্রাকচার নেমস্পেস গঠনের উপায় হচ্ছে প্যাকেজ। 'ডটেড মডিউল নেম' পদ্ধতি ব্যবহার করে প্যাকেজিং করা হয়। যেমন- A, B বলতে বুঝাচ্ছে- B হচ্ছে A-এর একটি সাব-মডিউল। এভাবে স্ট্রাকচার তৈরি করার কারণে এক মডিউলের সাথে আরেক মডিউলের গ্লোবাল ভ্যারিয়েবল নিয়ে বামেলা হওয়ার আশঙ্কা থাকে না।

এই প্যাকেজ ইম্পোর্ট করার সময় পাইথন sys.path ডিরেক্টরিতে সাব-প্যাকেজের জন্য অনুসন্ধান চালায়। উল্লেখ্য, __init__.py ফাইলটি থাকতেই হবে। এখন এই প্যাকেজ কীভাবে ব্যবহার করা যায়, তার কিছু উদাহরণ দেয়া যাক-

```
>>> import
sound.effects.echo
>>> sound.effects.echo.ech
ofilter(input, output,
delay=0.7, atten=4)
```

এভাবে ইম্পোর্ট করলে ফাংশন কল করার সময় পুরো মডিউলের নাম লিখে তারপর ফাংশনের নাম দিয়ে প্যারামিটার দিতে হবে। তাই এটা অন্যভাবেও লেখা যায়-

```
>>> from sound.effects
import echo
>>> echo.echofilter(input,
output, delay=0.7, atten=4)
```

তাহলে শুধু একটি সাব-মডিউল ইমপোর্ট এবং এর ফাংশনগুলো ব্যবহার করা হয়েছে। চাইলে শুধু ফাংশনগুলো ইম্পোর্ট করতে পারবেন। যেমন-

```
>>> from
sound.effects.echo import
echofilter
>>> echofilter(input, out-
put, delay=0.7, atten=4)
```

আবার একই মডিউলের মধ্যে যদি কোনো সাব-মডিউল বা ফাংশন ইম্পোর্ট করতে চান, তাহলে বেশ কয়েকটি উপায়ে তা করতে পারেন-

```
>>> from . import echo
>>> from .. import formats
>>> from ..filters import
equalizer
```

তবে মনে রাখতে হবে WU (.)-এর পরিমাণ নির্ভর করবে কোন মডিউল থেকে কোথায় ইম্পোর্ট করা হবে তার ওপরে

ফিডব্যাক : ahmadalsajid@gmail.com