

Day 27



How import works in Python:

What Is import in Python?

import is used to load code from another module (file) so you can use its functions, variables, or classes.

Example: importing a module.

```
1 import math
2 print(math.sqrt(16))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS E:\Python\Day21-30\Day27> python main.py
4.0
```

Example: importing specific items

```
4 from math import sqrt, pi
5 print(sqrt(25))
6 print(pi)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS E:\Python\Day21-30\Day27> python main.py
5.0
3.141592653589793
```

Example: Import Everything (Not Recommended)

```
8 from math import *
9 |
```

Example: importing with alias

```
10 import math as m
11 print(m.factorial(5))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS E:\Python\Day21-30\Day27> python main.py
120
```

Example: using 'dir' to know the function inside the module.

```
13 import math
14 print(dir(math))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\Python\Day21-30\Day27> python main.py
['_doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'cbrt', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'exp2', 'expm1', 'fabs', 'factorial', 'floor', 'fma', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'sumprod', 'tan', 'tanh', 'tau', 'trunc', 'ulp']
```

Example: importing your own file(modules)

A.py:

```
A.py > ...
1 def greet():
2     print("Hello")
3
```

B.py:

```
B.py
1 import A
2 A.greet()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS E:\Python\Day21-30\Day27> python B.py
Hello
```

How Python Finds a Module (Import Search Path)

Python searches in this order:

1. Current directory
2. Built-in modules
3. Installed packages
4. Paths in sys.path

if __name__ == "__main__" in Python:

What Is `__name__`?

- `__name__` is a special built-in variable in Python
- It tells how the Python file is being executed

Key Values of `__name__`:

How file is run	<code>__name__</code> value
Directly executed	"__main__"
Imported as a module	Module name (e.g., "utils")

Why Use `if __name__ == "__main__"`?

- To control code execution
- Code inside this block runs only when file is executed directly
- Prevents code from running when file is imported

Let's us first understand the issue without it:

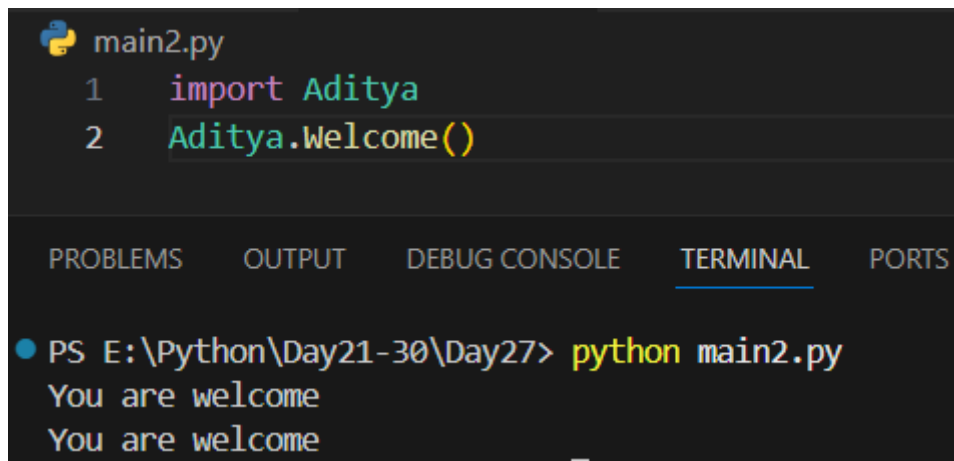
Aditya.py:

```
Aditya.py > ...
1  def Welcome():
2      print("You are welcome")
3  Welcome()

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\Python\Day21-30\Day27> python .\Aditya.py
You are welcome
```

Main2.py:



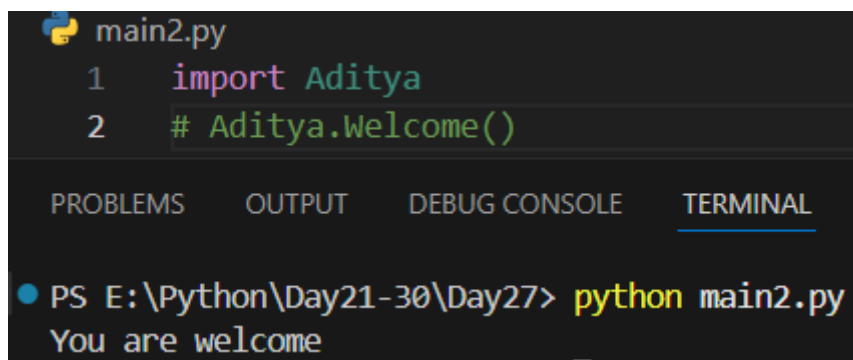
```
main2.py
1  import Aditya
2  Aditya.Welcome()

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● PS E:\Python\Day21-30\Day27> python main2.py
You are welcome
You are welcome
```

Clearly, when main2.py was run, it printed the output twice, once because of “Aditya.welcome()” and one because “welcome()” was called in Aditya.py.

What if we comment out “Aditya.welcome()”? Only once it will print.



```
main2.py
1  import Aditya
2  # Aditya.Welcome()

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● PS E:\Python\Day21-30\Day27> python main2.py
You are welcome
```

But, is this the solution? No, it means that every function inside the Aditya.py will get executed if main2.py was run.

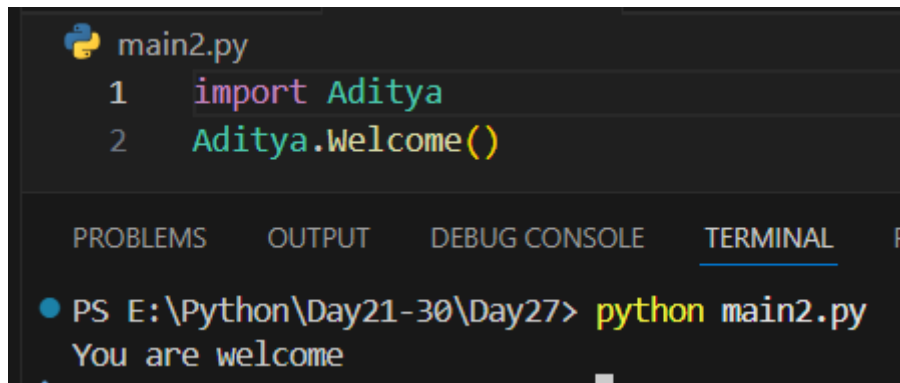
Solution:



```
Aditya.py > ...
1  def Welcome():
2      print("You are welcome")
3
4  if __name__ == "__main__":
5      Welcome()

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● PS E:\Python\Day21-30\Day27> python .\Aditya.py
You are welcome
```



The image shows a code editor window with a dark background. At the top, a file named 'main2.py' is open, containing two lines of Python code: `1 import Aditya` and `2 Aditya.Welcome()`. Below the code editor, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is active, showing a command prompt window. The prompt is `PS E:\Python\Day21-30\Day27>`, and the command `python main2.py` has been executed, resulting in the output `You are welcome`.

```
main2.py
1  import Aditya
2  Aditya.Welcome()

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  P

PS E:\Python\Day21-30\Day27> python main2.py
You are welcome
```

Real-World Use Case

- Python scripts often contain functions, classes, and test code
- Use `if __name__ == "__main__":` to run test/demo code only when needed

--The End--