

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.

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

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

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

A screenshot of a Python terminal window. At the top, there's a code editor with a single file named 'main.py' containing the following code:

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

Below the code editor are four tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is selected, indicated by an underline. In the terminal window, the command 'python main.py' is entered, followed by the output '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
```

A screenshot of a Python terminal window. The code editor shows:

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

The 'TERMINAL' tab is selected. The terminal output shows the results of running the script: '5.0' and '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.

The screenshot shows a Jupyter Notebook interface with the following details:

- Code Cell:** The code cell contains:

```
13 import math
14 print(dir(math))
```
- Output Cell:** The output cell displays the list of available methods and functions from the `math` module:

```
PS E:\Python\Day21-30\Day27> python main.py
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'cbtrt', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'exp2', 'expmt', 'fabs', 'factorial', 'floor', 'fma', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrtn', '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']
```
- Bottom Status Bar:** The status bar at the bottom shows the current working directory as "E:\Python\Day21-30\Day27" and indicates that a command-line interface (powershell) is active.

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__:

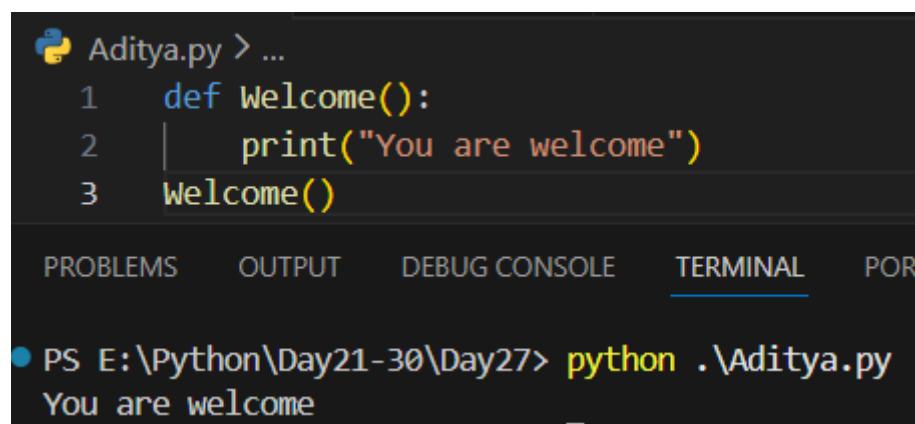
How file is run	__name__ value
Directly executed	"__main__"
Imported as a module	Module name (e.g., "utils")

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:

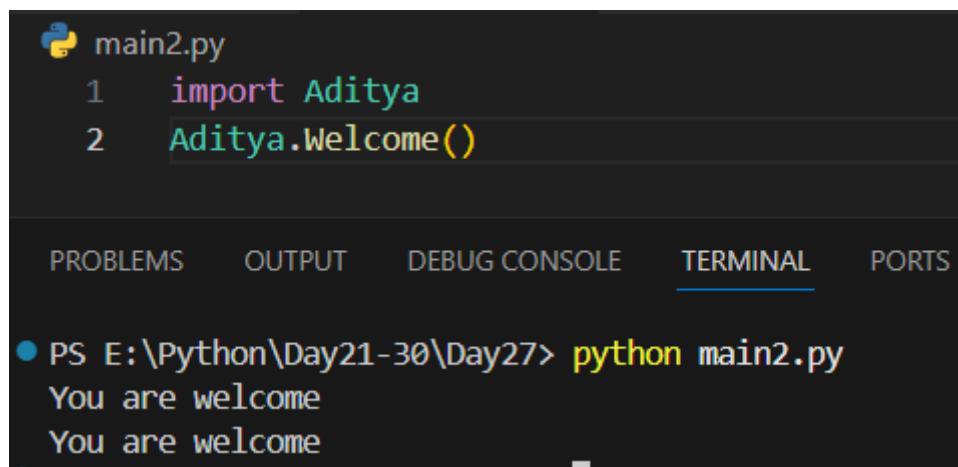


```
Aditya.py > ...
1 def Welcome():
2     print("You are welcome")
3 Welcome()

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORT
```

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

Main2.py:



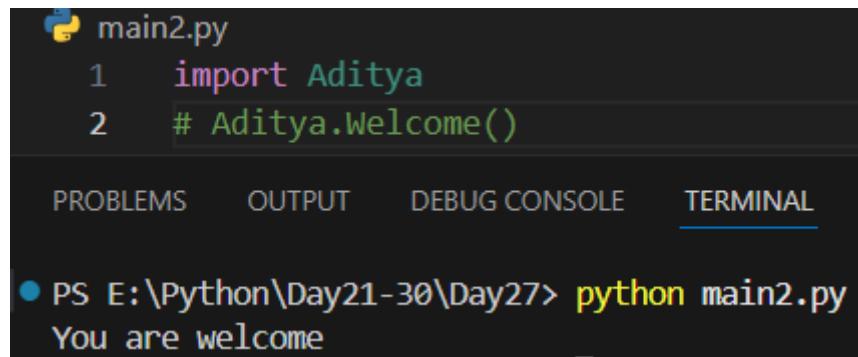
```
main2.py
1 import Aditya
2 Aditya.Welcome()

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

● 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 screenshot shows a Python code editor interface. On the left, there is a file tree icon followed by the filename "main2.py". The code editor contains the following Python script:

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

Below the code editor, there is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and P. The TERMINAL tab is currently selected, indicated by an underline. In the terminal window, the command "python main2.py" is entered, and the output "You are welcome" is displayed.

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--