

Overview

In this lesson, you will learn how to import and use **built-in Python modules** such as `math`, `turtle`, and `tkinter`. You will also learn that not all modules come installed with Python, how to recognize built-in libraries versus external ones, and how to find and use documentation to learn new modules effectively.

Important Information

Python includes many **built-in modules** that add extra functionality without you needing to write everything from scratch.

A **module** is a file of prewritten code that you can import and use in your program.

Examples of common built-in modules:

- `math` for mathematical operations
- `turtle` for drawing graphics
- `tkinter` for building simple graphical user interfaces (GUIs)

These modules are included with most standard Python installations.

Importing a Built-in Module

To use a built-in module, you must import it.

```
import math
```

Once imported, you access its functions using the module name as a namespace.

```
print(math.sqrt(25))
```

This keeps your code organized and makes it clear where functions come from.

Importing With an Alias

Some module names are long or used frequently. You can shorten them using `as`.

```
import math as m

print(m.sqrt(16))
```

This does not change how the module works—it only changes how you refer to it.

Importing Specific Tools From a Module

You can import only what you need from a module.

```
from math import sqrt, pi

print(sqrt(9))
print(pi)
```

In this case you do not need to use `math` as a prefix for functions. You also only have access to `sqrt()` and `pi`, but not any other functions or variables from the `math` module.

This can make code cleaner, but it is less obvious where functions come from.

Example: Using the `turtle` Module

`turtle` is a built-in module used for drawing and visual programming.

```
import turtle
t = turtle.Turtle()
t.forward(100)
t.left(90)
t.forward(100)

# Keep the window open
turtle.done()
```

Example: Using the `tkinter` Module

`tkinter` is Python’s standard library for creating basic windows.

```
import tkinter as tk

window = tk.Tk()
window.title("My First Window")

label = tk.Label(window, text="Hello, tkinter!")
label.pack()

window.mainloop()
```

Not All Modules Are Installed

Important distinction:

- **Built-in modules** come with Python
- **External libraries** must be installed separately

Examples of external libraries:

- `pygame`
- `numpy`
- `requests`

Installing external libraries requires extra tools and commands. This is **beyond the scope of this lesson** and will be covered later in the course.

For now:

- If `import` works without errors, the module is available
- If Python raises an `AttributeError` or `ModuleNotFoundError`, the module is not installed

Finding Documentation

You will often need to learn how a module works by reading documentation.

Good places to look:

- The official Python documentation
- Trusted tutorials
- Examples written by other programmers

A good search habit:

- Start with the module name and the word `python`
- Add the function name if you know it

Examples:

- “python math”
- “python turtle forward”
- “python tkinter Label”

Reading Documentation Effectively

When reading documentation:

- Look for **examples** first
- Pay attention to required imports
- Note parameter names and expected data types
- Skim headings before reading details

Good habits:

- Do not copy large blocks of code blindly
- Test small examples first
- Change values to see what happens

Common Errors and Fixes

`ModuleNotFoundError`

Cause: Module is not installed

Fix: Confirm whether the module is built-in or external

`AttributeError`

Cause: Function name is incorrect or does not exist

Fix: Check spelling and documentation

Nothing Happens

Cause: Required setup or loop is missing

Fix: Recheck example code carefully