The import Statement, math & random Modules

AiCE Software Engineering Foundations — Python Programming (Intermediate)

No professional developer writes everything from scratch. Python comes with an enormous library of pre-built tools — organized into modules — that you can bring into any program with a single line. This lesson opens that toolbox: what a module is, how to import one, and two of the most useful modules in Python's standard library.

1. What Is a Module?

When you're building a house in Abuja or Accra, you don't manufacture every material yourself — you go to the market and buy ready-made cement, iron rods, roofing sheets, and electrical fittings that other craftspeople have already produced. Your job is to bring them together and assemble something new.

A module in Python works exactly the same way. A module is a file containing ready-made Python code — functions, variables, and classes — that someone else has already written, tested, and packaged. Instead of writing that code yourself, you import the module into your program and use what's already there.

> **The Standard Library**
> *Python comes with a large collection of built-in modules called the Standard Library, installed automatically with Python — no downloads required. They cover mathematics, random number generation, date and time handling, file system interaction, networking, data compression, and much more. Beyond the Standard Library, hundreds of thousands of third-party modules exist, installed with a tool called pip — covered later in this program.*

2. The import Statement

Bringing a module into your program takes one line, conventionally placed at the very top of a file, before any other code:

```
import module_name
```

Once imported, you access the module's contents using dot notation:

```
module_name.function_name()
module_name.constant_name
```

The dot separates the module name from what you want to use inside it — the same dot notation you've already used with string methods, only here you're accessing tools from an imported module rather than methods on a string object.

3. The math Module

The math module provides mathematical functions and constants beyond Python's built-in arithmetic operators. You used it briefly in the beginner section — here it's covered thoroughly.

```
import math

# Mathematical constants
print(f"Pi         : {math.pi}")
print(f"Euler (e)  : {math.e}")

# Rounding functions
print(f"ceil(4.3)  : {math.ceil(4.3)}")
print(f"floor(4.9) : {math.floor(4.9)}")

# Power and roots
print(f"sqrt(25)   : {math.sqrt(25)}")
print(f"pow(2, 10) : {math.pow(2, 10)}")

# Logarithms
print(f"log10(100) : {math.log10(100)}")

# Trigonometry (values in radians)
print(f"sin(90°)   : {math.sin(math.radians(90)):.2f}")
```

The pattern is always the same: module name, dot, function or constant name. math.pi gives you pi, math.sqrt() calculates square roots, math.ceil() rounds up, math.floor() rounds down, and math.radians() converts degrees to radians for trigonometric functions.

Practical math Usage — Distance Calculator

A real example: calculating the straight-line distance between two African cities using the Pythagorean theorem:

```
import math

def straight_line_distance(dx_km, dy_km):
    """Calculate straight-line distance using Pythagorean
theorem."""
    return round(math.sqrt(dx_km**2 + dy_km**2), 2)

print(f"Lagos to Abuja  : {straight_line_distance(245, 430)}
km")
print(f"Nairobi to Accra: {straight_line_distance(3500, 280)}
km")
```

4. The random Module

The random module generates random numbers and makes random selections — essential for simulations, games, testing, shuffling data, and anywhere you need unpredictable variation.

```
import random

# Random integer between two values (inclusive)
print("Random integers (1-10):")
for _ in range(5):
    print(f"  {random.randint(1, 10)}")

# Random float between 0 and 1
print(f"Random float: {random.random():.4f}")

# Random choice from a list
african_cities = ["Lagos", "Nairobi", "Accra", "Kigali",
"Dakar"]
print(f"Random city: {random.choice(african_cities)}")

# Random sample (multiple unique choices)
sample = random.sample(african_cities, 3)
```

```
print(f"Random sample: {sample}")

# Shuffle a list in place
scores = [78, 85, 92, 67, 88]
random.shuffle(scores)
print(f"Shuffled: {scores}")
```

Every run of this code produces different output — that's the nature of randomness. Run it several times and observe the variation.

Function	What It Does	Example
random.randint(a, b)	Random integer between a and b (inclusive)	random.randint(1, 100)
random.random()	Random float between 0.0 and 1.0	random.random()
random.uniform(a, b)	Random float between a and b	random.uniform(1.5, 9.5)
random.choice(seq)	One random item from a sequence	random.choice(cities)
random.sample(seq, k)	k unique random items from a sequence	random.sample(cities, 3)
random.shuffle(seq)	Shuffles a list in place (modifies original)	random.shuffle(scores)

Practical random Usage — Student Lottery System

```
import random

def run_scholarship_lottery(applicants, num_winners):
    """Randomly select scholarship winners from a list of
applicants."""
    if num_winners > len(applicants):
        return None, "Not enough applicants for that many
winners."
    winners = random.sample(applicants, num_winners)
    return winners, None

applicants = ["Amara Diallo", "Kofi Mensah", "Ngozi Adeyemi",
```

```
              "Fatima Al-Hassan", "Kwame Asante", "Zainab
Musa"]

winners, error = run_scholarship_lottery(applicants, 3)
if error:
    print(f"Error: {error}")
else:
    print("AiCE Scholarship Lottery Winners:")
    for i, winner in enumerate(winners, 1):
        print(f"  {i}. {winner}")
```

Reproducible Randomness with random.seed()

By default, random uses the system clock as a seed, producing different results every run. If you need reproducible results — for testing or debugging — call random.seed(n) with a fixed number before using any random functions. The same seed always produces the same sequence of random values.

Key Takeaways

A module is a pre-built file of Python code you bring in with import module_name, then access with module_name.thing — dot notation, exactly as you've used with strings. The math module covers constants and calculations beyond basic arithmetic; the random module covers everything from single random numbers to shuffling entire lists, with random.seed() available whenever you need reproducible results for testing. The next lesson covers two more essential standard library modules: datetime for dates and times, and os for interacting with the file system from inside a Python program.