Built-In Math Functions & Number Formatting

AiCE Software Engineering Foundations — Python Programming

Arithmetic operators handle calculations, but real programs also need to round, measure, total, and display numbers cleanly. This lesson covers Python's built-in math functions, the math module for more advanced operations, and the formatting tools that turn raw numbers into professional-looking output.

1. Built-In Math Functions

Python provides several built-in functions for common mathematical operations — available immediately, with nothing to import.

abs() — Absolute Value

abs() returns a number's distance from zero, always positive:

```
print(abs(-45))       # 45
print(abs(45))        # 45
print(abs(-3.14))     # 3.14
```

Absolute values are used in distance calculations, error measurements, and anywhere you need a number's magnitude regardless of its sign.

round() — Rounding Numbers

round() rounds a number to a specified number of decimal places, or to the nearest whole number if none is specified:

```
price = 4750.678
print(round(price))           # 4751 — nearest integer
print(round(price, 2))        # 4750.68 — 2 decimal places

score = 84.5
print(round(score))           # 85 — rounds up at .5
```

round() Matters for Professional Output

Showing a price as 395187.5000000001 due to floating-point precision issues looks unprofessional. round() gives you control over exactly how many decimal places appear, which matters constantly for currency, percentages, and scientific values.

max() and min() — Finding Extremes

```
scores = [78, 92, 85, 67, 95, 88, 72, 91]

print(f"Highest score: {max(scores)}")
print(f"Lowest score : {min(scores)}")

# Or pass values directly
print(max(150, 200, 175, 225, 190))    # 225
print(min(150, 200, 175, 225, 190))    # 150
```

sum() — Totalling a Collection

```
monthly_revenue = [450000, 520000, 610000, 490000, 580000,
700000]
total = sum(monthly_revenue)
average = total / len(monthly_revenue)

print(f"Total 6-month revenue: {total} NGN")
print(f"Monthly average      : {round(average, 2)} NGN")
```

The Four Most Common Numerical Functions

sum(), max(), min(), and len() are the four most frequently used built-in functions on numerical collections — you'll see them constantly in data processing, analytics, and reporting code, including in the lists lesson ahead.

2. The math Module — Advanced Mathematical Functions

For more advanced mathematics — square roots, ceiling/floor, powers, constants — Python provides the math module. A module is a collection of pre-built functions brought into your program with the import statement:

```
import math

print(f"Pi: {math.pi}")
print(f"Square root of 144: {math.sqrt(144)}")

value = 4.3
print(f"ceil(4.3) = {math.ceil(value)}")       # 5
print(f"floor(4.3) = {math.floor(value)}")     # 4
```

import math at the top of the file brings the entire module in; you then access its functions with math.function_name() notation. The math module contains dozens of functions — sqrt, log, sin, cos, tan, factorial, and many more.

> **Modules Are Pre-Built Tools**
> *Importing a module is like pulling a toolbox off a shelf — the tools are already built, you just declare you want to use them. Python ships with many built-in modules: math, random, datetime, os, and more. External modules, installed with pip, expand this further.*

3. Number Formatting in Output

Displaying numbers cleanly is a genuinely professional skill. F-strings support format specifiers that control exactly how a number appears:

```
price = 4500.5
print(f"Price: {price:.2f}")                  # 4500.50

revenue = 24000000
print(f"Revenue: {revenue:,}")                # 24,000,000
```

```
completion_rate = 0.847
print(f"Completion: {completion_rate:.1%}")   # 84.7%
```

Specifier	Meaning	Example
:.2f	Fixed 2 decimal places	4500.50
:.0f	No decimal places (rounded)	4501
:,	Thousands separator	24,000,000
:,.2f	Thousands separator + 2 decimals	24,000,000.00
:.1%	Percentage with 1 decimal place	84.7%
:<15	Left-aligned in a 15-character field	text padded right
:>8	Right-aligned in an 8-character field	text padded left

The alignment specifiers are especially useful for building aligned table output:

```
print(f"{'Student':<15} {'Score':>8}")
print(f"{'Amara Diallo':<15} {92:>8}")
print(f"{'Kofi Mensah':<15} {78:>8}")
```

This produces neatly aligned columns — exactly the kind of clean terminal output real reporting tools rely on.

4. Bringing It All Together — The AiCE Financial Calculator

A complete program combining arithmetic, formatting, and the math module:

```
import math

fee_per_student = 85000
total_students = 1200
total_revenue = fee_per_student * total_students
```

```
instructor_costs = 12000000
platform_costs = 8500000
total_expenses = instructor_costs + platform_costs

net_profit = total_revenue - total_expenses
profit_margin = net_profit / total_revenue

print(f"Total Revenue   : NGN {total_revenue:,}")
print(f"Net Profit      : NGN {net_profit:,}")
print(f"Profit Margin   : {profit_margin:.1%}")
print(f"Sqrt of Students: {math.sqrt(total_students):.2f}")
```

The output is a fully formatted financial summary — the kind of terminal report you'd find in a real business application, combining arithmetic, number formatting, and Python's math tools in one program.

Key Takeaways

abs(), round(), max(), min(), and sum() cover the everyday math operations you'll reach for constantly, while the math module — imported once with import math — handles everything more advanced, from square roots to constants. F-string format specifiers like :.2f, :,, and :.1% are what separate raw numbers from genuinely professional output. With numbers and arithmetic covered, the next lesson brings numbers and strings together in a new context: user input, and making your programs genuinely interactive.