PYTHON NOTES: ARITHMETIC OPERATORS

A concise reference for Python's arithmetic operators with examples and practical tips.

CREATION CHECKLIST (WHAT I DID)

- Summarized definitions of numeric types (int and float).
- Compiled a clean, easy-to-scan operator reference table with examples and meanings.
- Added runnable Python examples covering each operator.
- Highlighted important behavior notes (division, floor division, parity checks, exponentiation).
- Listed everyday practical uses and patterns.
- Formatted for readability with headings, lists, a table, and code blocks.

WHAT ARE ARITHMETIC OPERATORS?

Arithmetic operators are symbols used to perform mathematical operations on numbers in Python.

Python numeric types include:
int → whole numbers (e.g., 10)
float → decimal numbers (e.g., 5.6)

OPERATOR REFERENCE

Operator	Symbol	Example	Meaning
Addition	**+**	5 + 3 → 8	Adds values
Subtraction	**-**	9 - 4 → 5	Subtracts values
Multiplication	*****	6 * 4 → 24	Multiplies values
Division	**/**	9 / 2 → 4.5	True division (result is float)
Floor Division	**//**	9 // 2 → 4	Quotient with decimal discarded (floored)

Operator	Symbol	Example	Meaning
Modulus	**%**	9 % 2 → 1	Remainder
Exponent	****	2 ** 3 → 8	Power (x to the power y)

EXAMPLES IN PYTHON

```
a = 10
b = 3

print(a + b)   # Addition  → 13
print(a - b)   # Subtraction → 7
print(a * b)   # Multiplication → 30
print(a / b)   # Division → 3.3333333333333335 (float)
print(a // b)  # Floor division → 3
print(a % b)   # Modulus → 1
print(a ** b)  # Exponent → 1000
```

IMPORTANT NOTES

- **Division vs. Floor Division**
  10 / 2 → 5.0 (float)
  10 // 2 → 5 (int result value; type may be int)
- **Parity (even/odd)** using modulus:
  `num % 2 == 0` → even
  `num % 2 == 1` → odd
- **Exponentiation (**)** common uses:
  Square → `num ** 2`
  Cube → `num ** 3`

PRACTICAL USES

Use	Operator(s)	Example
Calculating bill, salary	`+ - * /`	`total = subtotal + tax - discount`
Checking even/odd	`%`	`is_even = (n % 2 == 0)`

Use	Operator(s)	Example
Getting last digit	`%`	`last_digit = n % 10`
Power calculations	`**`	`area = r ** 2 * 3.14159`
Removing decimals (age, floors)	`//`	`years = days // 365`

QUICK TIPS

- Use `//` when you need an integer-like quotient (e.g., batching items into fixed-size boxes).
- Use `%` with `//` together to split into quotient and remainder: `q, r = n // d, n % d`.
- Beware integer overflow is not an issue in Python ints; they expand to big integers automatically.
- Floating-point has precision limits; for money, consider `decimal.Decimal`.

QUESTIONS & NEXT STEPS

Would you like this turned into a printable PDF or enriched with practice exercises and solutions?