Operator Precedence in Python: Who Goes First?

Hey Python rule-breakers! Today, we're tackling **operator precedence**—the order Python uses to evaluate expressions with multiple operators. It's like math's PEMDAS, but Python-style! We'll follow this table (high to low precedence):

Operators	Meaning
()	Parentheses
**	Exponent
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
<<, >>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
==, !=, >, >=, <, <=, is, is not, in, not in	Comparison, Identity, Membership
not	Logical NOT
and	Logical AND
or	Logical OR

Run the cells, tweak the expressions, and try the challenges—let's see who gets priority!

Why Precedence Matters

Without precedence, `2 + 3 * 4` could be `(2 + 3) * 4 = 20` or `2 + (3 * 4) = 14`. Python picks the second (14) because `*` beats `+`. Let's explore each level!

```python
# Example 1: Parentheses Rule All
print("No parens: 2 + 3 * 4 =", 2 + 3 * 4)        # 14 (* first)
print("With parens: (2 + 3) * 4 =", (2 + 3) * 4)  # 20 (+ first)
```

```python


# Challenge: Force the order!
a = 5
b = 2
c = 3
result = a + b * c  # Normal precedence
your_result = (a + b) * c  # Force with parens
print("Normal:", result)
print("Your parens:", your_result)
```

```python
# Example 2: Exponent Beats Others
print("2 ** 3 * 4 =", 2 ** 3 * 4)      # 32 (2^3 = 8, then * 4)
print("(2 ** 3) * 4 =", (2 ** 3) * 4)  # Same, explicit
```

```python
# Challenge
base = int(input("Enter a base: "))
exp = 2
mult = 3
power_result = base ** exp * mult
print(f"{base} ** {exp} * {mult} =", power_result)
```

```python
# Example 3: Multiplicative Group
print("10 * 2 + 3 =", 10 * 2 + 3)      # 23 (* first)
print("10 + 2 * 3 =", 10 + 2 * 3)      # 16 (* first)
print("10 / 2 - 1 =", 10 / 2 - 1)      # 4.0 (/ first)
print("7 // 2 * 3 =", 7 // 2 * 3)      # 9 (// then *)
print("10 % 3 + 1 =", 10 % 3 + 1)      # 2 (% then +)
```

```python
# Left-to-right within same level
print("10 / 2 * 3 =", 10 / 2 * 3)      # 15.0 (left to right)
```

```python
# Challenge
x = int(input("Enter a number: "))
result = x * 2 / 4 + 1
print(f"{x} * 2 / 4 + 1 =", result)
```

```
Start coding or generate with AI.
```

```python
# Example 4: Additive Group
print("5 + 3 * 2 =", 5 + 3 * 2)        # 11 (* first)
print("5 - 2 + 1 =", 5 - 2 + 1)        # 4 (left to right)
```

```python
# Challenge: Add or subtract!
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
c = a + b - 2 * 3
print(f"{a} + {b} - 2 * 3 =", c)
```

Bitwise Shift (<<, >>)

```
# Example 5: Bitwise Shifts
print("2 << 1 + 1 =", 2 << 1 + 1)      # 8 (+ first, then <<)
print("(2 << 1) + 1 =", (2 << 1) + 1)  # 5 (<< first)
# 2 << 1 = 4 (shift left), then + 1
```

```
print("8 >> 2 - 1 =", 8 >> 2 - 1)      # 4 (- first, then >>)
```

```
# Challenge:
num = int(input("Enter a number: "))
shift_result = num << 2 + 1
print(f"{num} << 2 + 1 =", shift_result)
```

Bitwise AND (&)

```
# Example 6: Bitwise AND
print("5 & 3 + 1 =", 5 & 3 + 1)        # 1 (+ first, then &)
print("(5 & 3) + 1 =", (5 & 3) + 1)    # 2 (& first)
# 5 & 3 = 1 (binary 101 & 011 = 001)
```

```
# Challenge: AND it!
x = int(input("Enter a number: "))
result = x & 2 + 4
print(f"{x} & 2 + 4 =", result)
# Try (x & 2) + 4!
```

Bitwise XOR (^)

```
# Example 7: Bitwise XOR
print("6 ^ 2 * 3 =", 6 ^ 2 * 3)        # 0 (* first, then ^)
print("(6 ^ 2) * 3 =", (6 ^ 2) * 3)    # 12 (^ first)
# 6 ^ 2 = 4 (binary 110 ^ 010 = 100)
```

```
# Challenge: XOR it!
y = int(input("Enter a number: "))
result = y ^ 3 - 1
print(f"{y} ^ 3 - 1 =", result)
# Try y ^ (3 - 1)!
```

Comparison, Identity, Membership

```
# Example 8: Comparisons and More
print("5 > 2 + 1 =", 5 > 2 + 1)        # True (+ first)
print("3 == 4 - 1 =", 3 == 4 - 1)      # True (- first)
print("2 in [1, 2] and 3 > 1 =", 2 in [1, 2] and 3 > 1)  # True (in, >, then and)
```

```
# Identity (is)
a = [1]
b = a
print("a is b + [] =", a is b + [])    # False (+ first)
```

```
# Challenge: Compare it!
num = int(input("Enter a number: "))
check = num > 5 - 2
print(f"{num} > 5 - 2 is", check)
# Try num != 3 * 2 or "a" in "cat"!
```

Logical NOT (not)

```
# Example 9: Logical NOT
print("not 2 + 1 > 3 =", not 2 + 1 > 3)  # True (+, >, then not)
print("not (2 + 1 > 3) =", not (2 + 1 > 3))  # True (parens first)
```

```
# Challenge: Flip it!
val = int(input("Enter a number: "))
result = not val * 2 > 5
print(f"not {val} * 2 > 5 is", result)
```

Logical AND (and)

```
# Example 10: Logical AND
print("2 > 1 and 3 < 4 =", 2 > 1 and 3 < 4)  # True (>, <, then and)
print("5 - 2 > 1 and 2 * 2 < 5 =", 5 - 2 > 1 and 2 * 2 < 5)  # True
```

```
# Challenge: AND it!
x = int(input("Enter a number: "))
result = x + 1 > 0 and x * 2 < 10
print(f"{x} + 1 > 0 and {x} * 2 < 10 is", result)
# Try x > 3 and x < 5!
```

Logical OR (or)

```
# Example 11: Logical OR
print("2 > 3 or 4 < 5 =", 2 > 3 or 4 < 5)    # True (>, <, then or)
print("not 2 + 1 > 3 or 5 - 2 < 4 =", not 2 + 1 > 3 or 5 - 2 < 4)  # True
```

```

# Challenge
y = int(input("Enter a number: "))
result = y < 0 or y > 5 + 2
print(f"{y} < 0 or {y} > 5 + 2 is", result)
# Try y == 3 or y * 2 > 4!
```

```
Enter a number: 5
5 < 0 or 5 > 5 + 2 is False
```

```
Start coding or generate with AI.
```