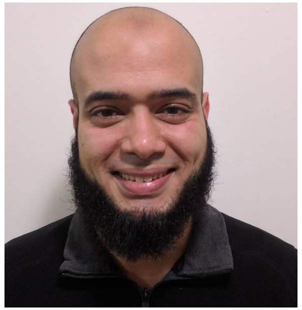
Python Programming
Operator Associativity

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Operator Associativity

- What if operators have **the same priority**? E.g. + -
  - Associativity: group either from left or from right
- Let’s say we have expression: 10 - 6 + 3
- Left-to-right associativity: **group** from left to right
  - **(10 - 6)** + 3 ⇒ 4 + 3 = 7
  - **7-6**+5-4+3-2+1 ⇒ **1+5**-4+3-2+1 ⇒ **6-4**+3-2+1 ⇒ **2+3**-2+1 ⇒**5-2**+1 ⇒**3+1** ⇒4
- Right-to-Left associativity: **group** from right to left
  - 10 - **(6 + 3)** ⇒ 10 - 9 = 1   [wrong!]
  - 7-6+5-4+3-**2+1** ⇒ 7-6+5-4+**3-3** ⇒ 7-6+5-**4+0** ⇒ 7-6+**5-4** ⇒ 7-**6+1** ⇒ **7-7** ⇒ 0

Operator Associativity

- Almost all the operators have left-to-right associativity
    - 10 - 6 + 3 ⇒ (10 - 6) + 3 ⇒ 4 + 3 = 7
- Exponent operator ** has **right-to-left associativity** in Python
    - 2 ** 3 ** 4 ⇒ 2 ** (3 ** 4) = 2 ** 81 = 2417851639229258349412352

Non associative operators

- X = Y = Z = 3
  - Is this left to right associativity? ((X = Y) = Z) = 3 ?
  - Is this right to left associativity? (X = (Y = (Z = 3)))
  - Both doesn’t make sense
- = does not have associativity
  - X = Y = Z = 3 is just implemented to assign all to value 3
- More technically: *Since assignments are statements, not operations, the assignment operator does not have a value and is not associative*
  - *2 + 3 has value 5*
  - *X = 2 doesn’t have a value: it assigns 2 to X*

Order of Evaluation

- (2 ** 10) / (2 + 3 * 4)
  - Which expression will be evaluated first? (2 ** 10) ? (2 + 3 * 4)?
  - The **left** operand is always evaluated before the right operand
    - Same for function arguments
- (1+2) ** (3-1) ** (4-2)
  - **Left to right** evaluation for every expression
  - **Right to left** associativity to compute final results
- Coming from C++? No order guarantee

Precedence vs Associativity vs Order of Evaluation

- **Operator precedence** specifies the **order of operations** in expressions that contain more than one operator (e.g. * before +)
- **Associativity** is about how to **group operands** (if operations has the same priority),
  - But **first**, we need to evaluate operands/subexpressions
- **Order of evaluation** is about the **order** of evaluating the operands
  - Always left first

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."