



Operator, Operands, dan Expressions

Setiap logika komputasi yang berada dalam program Anda akan menggunakan ekspresi. Sebuah ekspresi dapat terdiri dari operator dan operan. Salah satu contoh termudah adalah $a + b$ atau $2 + 3$, yang dapat kita pecah yakni $+$ sebagai operator dan a , b , 2 , atau 3 sebagai variabel/operand. Operator jelas memiliki fungsinya masing-masing dan direpresentasikan dengan simbol atau keyword tertentu.

Tip: Anda dapat melakukan expression tanpa menggunakan variabel secara langsung pada mode interaktif Python:

```
1. print(2 + 3)
```

Output:

5

```
1. print(3 * 5)
```

Output:

15

Jenis-jenis operator

Simak di bawah ini beberapa operator yang ada di Python.

Matematika dan String

- $+$ (tambah)
 - Menambahkan dua objek.
 - $3 + 5$ menghasilkan 8
 - $'a' + 'b'$ menghasilkan $'ab'$.
- $-$ (kurang)



DIBANTU



- -5.2 adalah expression yang sama dengan 0 - 5.2 menghasilkan -5.2.
- 50 - 24 menghasilkan 26.
- Tidak berlaku untuk string, akan menghasilkan error `unsupported operand`.
- * (perkalian)
 - Mengembalikan hasil perkalian angka atau mengembalikan string yang diulang sejumlah tertentu.
 - $2 * 3$ menghasilkan 6.
 - `'la' * 3` menghasilkan `'lalala'`.
- ** (pangkat)
 - Mengembalikan operand pertama pangkat operand kedua.
 - $3 ** 4$ menghasilkan 81 (sama dengan $3 * 3 * 3 * 3$).

| Tips: untuk akar dua, gunakan pangkat 0.5.
- / (pembagian)
 - Mengembalikan hasil pembagian operand pertama dengan operand kedua (float).
 - $13 / 3$ menghasilkan 4.333333333333333.
- // (pembagian habis dibagi / div)
 - Mengembalikan hasil pembagian operand pertama dengan operand kedua (bilangan bulat), kecuali jika salah satu operand adalah float, akan menghasilkan float.
 - $13 // 3$ menghasilkan 4.
 - $-13 // 3$ menghasilkan -5.
 - $9 // 1.81$ menghasilkan 4.0.
- % (modulo)
 - Mengembalikan sisa bagi.
 - $13 \% 3$ menghasilkan 1.
 - $-25.5 \% 2.25$ menghasilkan 1.5.

Operasi Bit

- << (left shift)
 - Menggeser representasi bit/binary dari operand pertama sebanyak operand kedua ke kiri.
 - $2 << 2$ menghasilkan 8.
 - 2 direpresentasikan sebagai 10 dalam binary.
 - Geser ke kiri sebanyak 2x, menjadi 1000 (tambahkan 0 di belakangnya).
 - 1000 dalam binary bernilai 8 dalam desimal.
- >> (right shift)



DIBANTU



- 11 direpresentasikan sebagai 1011 dalam binary.
- Geser ke kanan sebanyak 1x, menjadi 101.
- 101 dalam binary bernilai 5 dalam desimal.
- & (bit-wise AND)
 - Menjalankan operasi binary AND pada representasi operand pertama dan kedua.
 - 5 & 3 menghasilkan 1.
 - Representasi binary 5 adalah 101 dan representasi binary 3 adalah 011.
 - 101 and 011 bernilai 001.
 - 001 dalam desimal adalah 1.
- | (bit-wise OR)
 - Menjalankan operasi binary OR pada representasi operand pertama dan kedua.
 - 5 | 3 menghasilkan 7.
 - Representasi binary 5 adalah 101 dan representasi binary 3 adalah 011.
 - 101 or 011 bernilai 111.
 - 111 dalam desimal adalah 7.
- ^ (bit-wise XOR)
 - Menjalankan operasi binary XOR pada representasi operand pertama dan kedua.
 - 5 ^ 3 menghasilkan 6.
 - Representasi binary 5 adalah 101 dan representasi binary 3 adalah 011.
 - 101 xor 011 bernilai 110.
 - 110 dalam desimal adalah 6.
- ~ (bit-wise invert)
 - Menjalankan operasi binary invert pada representasi operand.
 - Nilai invert dari x adalah $-(x+1)$, menggunakan metode Two's Complement
 - ~5 menghasilkan -6.
 - Lebih lanjut mengenai Two's Complement dapat dibaca pada https://en.wikipedia.org/wiki/Two%27s_complement

Perbandingan

- < atau operator.lt (less than)
 - Menjalankan perbandingan apakah operand pertama lebih kecil dari operand kedua.
 - 5 < 3 menghasilkan False and 3 < 5 menghasilkan True.



DIBANTU



- > atau operator.gt (greater than)
 - Menjalankan perbandingan apakah operand pertama lebih besar dari operand kedua.
 - `5 > 3` menghasilkan True.
- <= atau operator.le (less than or equal to)
 - Menjalankan perbandingan apakah operand pertama lebih kecil atau sama dengan operand kedua.
 - `x = 3; y = 6;`
 - `x <= y` menghasilkan True.
- >= atau operator.ge (greater than or equal to)
 - Menjalankan perbandingan apakah operand pertama lebih besar atau sama dengan operand kedua.
 - `x = 4; y = 3;`
 - `x >= y` menghasilkan True.
- == atau operator.eq (equal to)
 - Menjalankan perbandingan apakah operand pertama sama dengan operand kedua.
 - `x = 2; y = 2;`
 - `x == y` menghasilkan True.
 - `x = 'str'; y = 'stR';`
 - `x == y` menghasilkan False.
 - `x = 'str'; y = 'str';`
 - `x == y` menghasilkan True.
- != atau operator.ne (not equal to)
 - Menjalankan perbandingan apakah operand pertama tidak sama dengan operand kedua.
 - `x = 2; y = 3;`
 - `x != y` returns True.

Penggunaan lt, gt, le, ge, eq, ne

Pada materi di atas, Anda telah belajar mengenai beberapa operator untuk perbandingan. Sebagai contoh, operator.lt, artinya less than, digunakan sebagai operator untuk membandingkan apakah operand pertama lebih kecil dari operand kedua. Begitu juga untuk operator.gt (greater than) yang digunakan sebagai perbandingan apakah operand pertama lebih besar dari operand kedua. Selain lt dan gt, ada juga jenis operator lainnya, yaitu, le, ge, eq, dan ne.

Mari kita implementasikan operator perbandingan tersebut (lt, gt, ge, eq, dan ne) pada kasus jumlah kelereng berwarna hijau dan kuning.

**DIBANTU**



```
2. hijau = 5
3. kuning = 10
4. print('Kelereng Hijau = {}'.format(hijau))
5. print('Kelereng Kuning = {}'.format(kuning))
6. for func in (lt, le, eq, ne, ge, gt):
7.     print('{}(hijau, kuning): {}'.format(func.__name__, func(hijau, kuning)))
```

Output:

```
Kelereng Hijau = 5
Kelereng Kuning = 10
lt(hijau, kuning): True
le(hijau, kuning): True
eq(hijau, kuning): False
ne(hijau, kuning): True
ge(hijau, kuning): False
gt(hijau, kuning): False
```

Boolean Operator

- not (boolean NOT)
 - Jika x bernilai True, fungsi akan mengembalikan nilai False.
 - Jika x bernilai False, fungsi akan mengembalikan nilai True.
 - x = True;
 - not x akan mengembalikan nilai False.
- and (boolean AND)
 - x AND y akan mengembalikan nilai False jika x bernilai False, atau fungsi akan mengembalikan nilai y.
 - x = False; y = True;
 - x AND y, Fungsi akan mengembalikan nilai False karena x bernilai False.
 - Dalam kasus ini, Python tidak akan mengevaluasi nilai y karena apapun nilai y tidak akan mempengaruhi hasil. Hal ini dinamakan short-circuit evaluation.
- or (boolean OR)
 - x OR y, Jika x bernilai True, fungsi akan mengembalikan nilai True, atau fungsi akan mengembalikan nilai dari y.
 - x = True; y = False;
 - x OR y, fungsi akan mengembalikan nilai True.
 - Dalam kasus ini, Python juga menggunakan short-circuit evaluation karena apapun nilai y tidak akan mempengaruhi hasil.



DIBANTU


Cara singkat menuliskan operasi

Jika Anda melakukan assignment kembali hasil sebuah expression, beberapa di antaranya bisa disingkat sebagai berikut:



```
1. a = 2
2. a = a * 3
```

Dapat dituliskan sebagai



```
1. a = 2
2. a *= 3
```

Perhatikan formatnya menjadi [operand] [operasi] = expression.

Urutan matematis dalam melakukan evaluasi

Jika Anda memiliki expression `2 + 3 * 4`, Apakah penambahan dilakukan terlebih dahulu sebelum perkalian? Jika merujuk pada aturan yang benar, maka perkalian dilakukan lebih dahulu, sehingga perkalian memiliki urutan lebih awal/tinggi.

Berikut adalah tabel urutan yang diambil dari referensi [Dokumentasi Python](#):

Operator	Description
lambda	Lambda expression
if - else	Conditional expression
or	Boolean OR
and	Boolean AND
not x	Boolean NOT
in, not in, is, is not, <, <=, >, >=, !=, ==	Comparisons, including membership tests and identity tests
	Bitwise OR
^	Bitwise XOR
&	Bitwise AND



<<, >>	Shifts
+, -	Addition and subtraction
*, @, /, //, %	Multiplication, matrix multiplication, division, floor division, remainder
+X, -X, ~X	Positive, negative, bitwise NOT
**	Exponentiation
await x	Await expression
x[index], x[index:index], x(arguments...), x.attribute	Subscription, slicing, call, attribute reference
(expressions...), [expressions...], {key: value...},{expressions...}	Binding or tuple display, list display, dictionary display, set display

Tip: Gunakan kurung untuk memudahkan keterbacaan dan memastikan urutan secara tepat. $2 + (3 * 4)$ akan lebih mudah dibaca daripada $2 + 3 * 4$, meskipun hasilnya sama

Gunakan kurung untuk mengubah urutan operasi. $2 + 3 * 4$ seharusnya dilakukan terlebih dahulu. Anda dapat menggunakan $(2 + 3) * 4$ untuk memastikan penjumlahan dilakukan terlebih dahulu.

Operator di Python juga bersifat asosiatif dari kiri ke-kanan. Artinya, semua operator yang memiliki tingkatan yang sama akan dijalankan berurutan dari kiri ke kanan.



Dicoding Space
Jl. Batik Kumeli No.50, Sukaluyu,
Kec. Cibeunying Kaler, Kota Bandung
Jawa Barat 40123

Penghargaan



Decode Ideas
Discover Potential
[Tentang Kami](#)

[Blog](#)
[Reward](#)
[Showcase](#)
[Hubungi Kami](#)
[FAQ](#)





© 2022 Dicoding | Dicoding adalah merek milik PT Presentologics, perusahaan induk dari PT Dicoding Akademi Indonesia.

[Terms](#) • [Privacy](#)



DIBANTU