📘 Materi: Bitwise Binary dalam Python

1. Pengantar

Bitwise berasal dari kata *bit* (digit biner: `0` dan `1`) dan *wise* (berdasarkan).
Artinya, **bitwise operation adalah operasi yang dilakukan langsung pada level bit**.

Komputer sebenarnya hanya mengerti bilangan biner (`0` dan `1`). Dengan memahami operasi bitwise, kita bisa: - Memahami cara kerja komputer di level dasar. - Melakukan perhitungan yang sangat cepat. - Menerapkan optimasi pada program (misalnya dalam kriptografi, jaringan, atau sistem embedded).

---

2. Representasi Biner di Python

Di Python, kita bisa melihat representasi biner sebuah angka dengan fungsi `bin()`.

```
# Representasi biner
a = 10
b = 4

print("a dalam biner:", bin(a))  # 10 -> 0b1010
print("b dalam biner:", bin(b))  # 4  -> 0b100
```

Output:

```
a dalam biner: 0b1010
b dalam biner: 0b100
```

👉`0b` adalah prefix (penanda) bahwa bilangan tersebut ditulis dalam format **biner**.

---

3. Operator Bitwise di Python

Python menyediakan beberapa operator bitwise:

Operator	Nama	Contoh	Penjelasan
`&`	AND	a & b	Bit bernilai `1` jika kedua bit sama-sama `1`
`|`	OR	a | b	Bit bernilai `1` jika salah satu bit `1`
`^`	XOR	a ^ b	Bit bernilai `1` jika hanya salah satu `1`
`~`	NOT (Komplemen)	~a	Membalik semua bit
`<<`	Shift Left	a << 2	Geser bit ke kiri (kali 2^n)

Operator	Nama	Contoh	Penjelasan
`>>`	Shift Right	a >> 2	Geser bit ke kanan (bagi 2^n)

---

4. Contoh Penggunaan Operator Bitwise

a) AND (`&`)

```
a = 10  # 1010
b = 4   # 0100

print("a & b:", a & b)
print("Biner:", bin(a & b))
```

Hasil:

```
a & b: 4  -> 0b100
```

👃 Hanya bit ke-3 yang sama-sama `1`.

---

b) OR (`|`)

```
print("a | b:", a | b)
print("Biner:", bin(a | b))
```

Hasil:

```
a | b: 14  -> 0b1110
```

👃 Bit `1` muncul jika salah satu `1`.

---

c) XOR (`^`)

```
print("a ^ b:", a ^ b)
print("Biner:", bin(a ^ b))
```

Hasil:

```
a ^ b: 14 -> 0b1110
```

👃 Hanya berbeda yang bernilai `1`.

---

d) NOT (`~`)

```
print("~a:", ~a)
print("Biner:", bin(~a))
```

Hasil:

```
~10 = -11
```

👃 NOT membalik semua bit, termasuk tanda bilangan (karena Python memakai representasi **two's complement**).

---

e) Shift Left (`<<`)

```
print("a << 1:", a << 1)   # Geser 1 kali
print("a << 2:", a << 2)   # Geser 2 kali
```

Hasil:

```
a << 1: 20
a << 2: 40
```

👃 Sama dengan mengalikan `a` dengan `2^n`.

---

f) Shift Right (`>>`)

```
print("a >> 1:", a >> 1)   # Geser 1 kali
print("a >> 2:", a >> 2)   # Geser 2 kali
```

Hasil:

```
a >> 1: 5
a >> 2: 2
```

👃 Sama dengan membagi `a` dengan `2^n`.

---

5. Studi Kasus Sederhana

🔒 Mengecek Apakah Bilangan Genap atau Ganjil

```python
def cek_genap_atau_ganjil(x):
    if x & 1 == 0:
        return "Genap"
    else:
        return "Ganjil"

print(cek_genap_atau_ganjil(10))  # Genap
print(cek_genap_atau_ganjil(7))   # Ganjil
```

👃 Dengan `x & 1`, kita bisa langsung cek bit terakhir (LSB) bilangan: - Jika `0` → Genap. - Jika `1` → Ganjil.

---

6. Ringkasan

- **Bitwise** = operasi langsung di level bit (`0` dan `1`).
- Operator utama: `&`, `|`, `^`, `~`, `<<`, `>>`.
- Sangat berguna untuk **optimasi, kriptografi, jaringan, pengolahan data biner, dan pemrograman low-level**.
- Python memudahkan kita melihat biner dengan fungsi `bin()`.