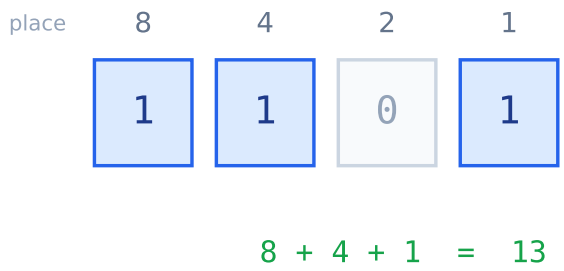
Data representation

Python Reference

Bits & binary

A bit 比特 is a single `0` or `1`. Binary 二进制 is the base-2 number system: each place is worth twice the one to its right (1, 2, 4, 8, …). Denary 十进制 (base-10) is our normal numbers.



binary: powers of 2 · bit k has weight 2^k

Binary place values: 1101 means 8 + 4 + 1 = 13

```
print(bin(13))         # 0b1101
print(int("1101", 2))  # 13
```

- 8 bits make a byte 字节. A fixed width can overflow 溢出 (wrap around) when the number is too big.

```
x = 250
x = (x + 10) % 256     # one byte wraps at 256
print(x)               # 4
```

Hexadecimal 十六进制 (base 16) is a compact way to read binary: one hex digit stands for exactly four bits. Python writes hex with `0x`:

```
print(hex(255))        # 0xff
print(0xFF)            # 255
print(int("ff", 16))   # 255
```


Compression

Compression 压缩 makes data smaller. Lossless 无损 compression keeps every bit, so you rebuild the original exactly. Lossy 有损 compression throws away detail — smaller but not exact — and is used for photos and music.

Run-length encoding 游程编码 is a simple lossless method: store a run 游程 (a repeat) as a count plus the value.

```
def rle(text):
    out = ""
    i = 0
    while i < len(text):
        run = 1
        while i + run < len(text) and text[i + run] == text[i]:
            run += 1
        out += str(run) + text[i]
        i += run
    return out

print(rle("AAAABBBCCD"))   # 4A3B2C1D
```

Common mistakes

- `n` bits store `2**n` different values, from `0` up to `2**n - 1`.
- Lossy compression throws away detail and cannot be undone; lossless can be reversed exactly.