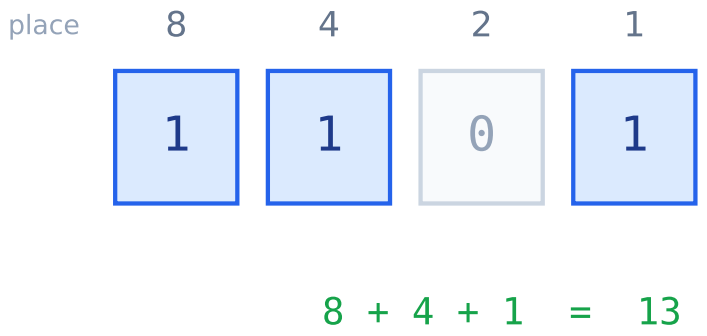
数据表示

Python Reference

比特与二进制

比特 (bit) 是单个的 `0` 或 `1`。二进制 (binary) 是以 2 为底的数制: 每一位的价值都是它右边一位的两倍 (1、2、4、8……)。十进制 (denary, 以 10 为底) 是我们平常用的数。



二进制位值:*1101* 表示 *8 + 4 + 1 = 13*

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

- 8 个比特组成一个字节 (byte)。当数字太大时, 固定的位宽会溢出 (overflow)(绕回)。

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

十六进制 (hexadecimal,base 16) 是阅读二进制的紧凑方式: 一个十六进制位恰好代表四个二进制位。Python 用 `0x` 表示十六进制:

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

数据压缩

压缩 (compression) 让数据变小。无损 (lossless) 压缩保留每一个比特, 所以你能精确还原原始数据。有损 (lossy) 压缩丢掉一些细节 — 更小但不精确 — 用于照片和音乐。

游程编码 (run-length encoding) 是一种简单的无损方法: 把一段游程 (run)(重复) 存成 “ 个数 + 值 ”。

```
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
```

常见错误

- `n` 个比特 (bit) 能存 `2**n` 种不同的值, 范围是 `0` 到 `2**n - 1`。
- 有损压缩 (lossy) 会丢掉细节且无法还原; 无损压缩 (lossless) 可以精确还原。