A = range(0,n)
sum = 0
i = 0
→ for x in A[i:]
    i += 1

⋮

$x \in (0, n-1)$ n
$x \in (1, n-1)$ $n-1$
$x \in (2, n-1)$ $n-2$
⋮
$x \in (n-1, n-1)$ 1

$$\sum_{n=1}^{n} i = \frac{n(n+1)}{2} = O(n^2)$$

for i in range(n):
    # do something

$O(n)$

---

i = 2.
while i < N:
    # do something
    i = i ** 2   # i^2

i = 2
i = 4
i = 16
i = 216
i = 46k

N = 10000
N = 100   3
N = 1000   4
10,000   4
46000
4-5

$O(\log_{10}(n))$

$\log n = \log_{10} n \approx c \log n$

$\log_x \quad \log_{10} \qquad \log_{10} x \qquad = O(\log_{10} n)$

```
for i in range(n):
    func_2(i)  # func2 is O(log n).
```

$O(n \log n)$

$O(n) \qquad O(n \log n)$

$$\frac{\cancel{n}}{\cancel{n} \log n} < 1$$

$$\frac{1}{\log(n)} < 1 \Rightarrow \log(n) > 1$$

$$2^{\log(n)} > 2^1$$

$$n > 2$$

$A = [x, y, z, a, b, c]$

... a in A :

find a

linear search is $O(n)$.

The complexity of linear search is **always** calculated as a lineal function.