# Algorithm design

Python Reference

## Algorithms & decomposition

An algorithm 算法 is a clear list of steps that solves a problem. Decomposition 分解 means breaking a big problem into smaller parts you can solve one at a time.

```
# Algorithm: find the largest number in a list
def largest(nums):
    best = nums[0]
    for n in nums:
        if n > best:
            best = n
    return best

print(largest([3, 9, 2, 7]))   # 9
```

- Abstraction 抽象 means ignoring detail: you use `largest(...)` without re-reading how it works.

## Pseudocode & flowcharts

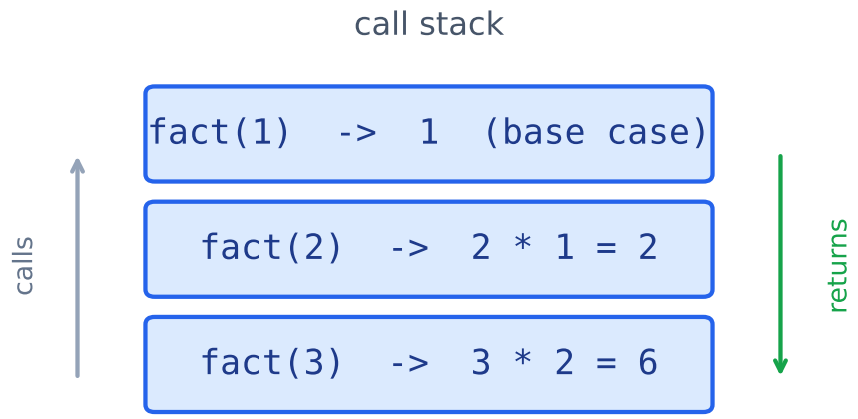
Pseudocode 伪代码 is plain, structured English for an algorithm, written before the real code. It is not run.

```
SET best TO first number
FOR each number n
    IF n > best THEN
        SET best TO n
OUTPUT best
```

A flowchart 流程图 draws the same plan: a box for each step, a diamond for each decision 判断, and arrows for the order.

## Recursion & the call stack

Recursion 递归 is when a function calls itself. It needs a base case 基准情形 (a simple input that returns at once) and a recursive case 递归情形 (it calls itself on a smaller input).




*The call stack for factorial(3): each call waits, then returns in reverse order*

```
def fact(n):
    return 1 if n <= 1 else n * fact(n - 1)

print(fact(5))      # 120
```

- Each paused call sits on the call stack 调用栈 until the call above it returns.