

Algorithm design

Python Reference

Algorithms & decomposition

An algorithm is a clear list of steps that solves a problem. Decomposition 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 means ignoring detail: you use `largest(...)` without re-reading how it works.

Pseudocode & flowcharts

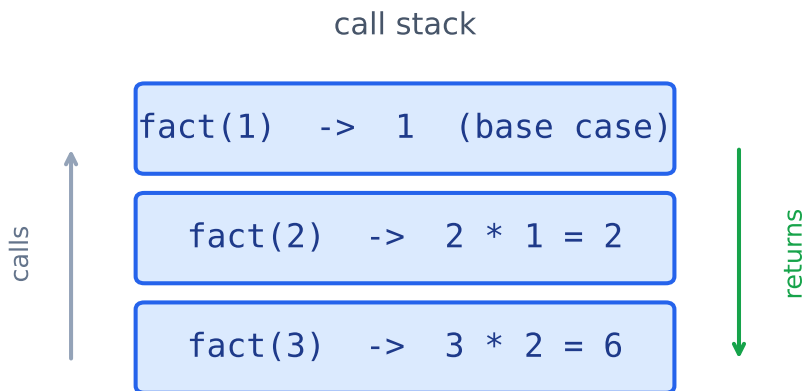
Pseudocode is plain, structured English for an algorithm, written before the real code. It is not run.

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

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

Recursion & the call stack

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



call stack: newest frame on top · returns unwind

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 until the call above it returns.

Common mistakes

- Recursion needs a base case, or it calls itself forever and crashes the call stack.
- Pseudocode is for planning — it need not run, but every step must be unambiguous.
- Break a big problem into small named steps before you write any code.