



a program runs top to bottom, until a decision or a loop sends it somewhere else



Sequence, the default flow

- without any decision or loop, Python runs every statement exactly once, top to bottom, in the order written
- a decision changes WHICH statements run, a loop changes HOW MANY TIMES a statement runs
- both still work by testing one condition, that evaluates to either True or False

for, counting through a known sequence

- for variable in sequence: runs the block once per item, binding the loop variable to each item in turn
- range(5) produces 0, 1, 2, 3, 4, five values, the stop value itself is never included
- range(2, 8) starts at 2 and stops before 8; range(0, 10, 2) adds a step, so it counts 0, 2, 4, 6, 8
- a for loop is the right choice whenever the number of passes is known in advance

if, elif, else *

- if runs its block only when the condition is True, and skips it otherwise
- elif adds another condition, checked only if every condition above it was False
- else needs no condition of its own, it catches whatever every if and elif above it missed
- the first True condition wins, every block below it is skipped even if it would also be True

$i = 2$ _____

$i = 1$ _____

$i = 0$ _____



three runs bottom to top, one loop variable value each pass; the drawn arrow links the bottom two runs, the single step that separates one pass from the next

One score, three grade bands

if marks ≥ 90 : elif marks ≥ 60 : else:

marks is 90 or more

marks is 60 to 89

neither condition holds

A grade

the first True condition, nothing below it runs

B grade

reached only because the first test failed

C grade

else needs no condition of its own

