Sequence, selection and iteration

6.2 · Problem solving with programming · revision notes

Sequence, selection and repetition are the building blocks of every program. You need to read, trace, complete and write Python code that uses if, elif and else, count-controlled for loops, condition-controlled while loops, and loops that visit every item in a list or string. They appear throughout Paper 2.

Key facts

- Sequence: statements run in the order written
- Selection: if / elif / else (only the first True branch runs)
- Count-controlled loop: for i in range(n): repeats n times
- Condition-controlled loop: while condition: repeats while the condition is True
- range(n) gives 0 to n − 1
- range(a, b) gives a to b − 1; range(a, b, s) goes up in steps of s
- Iteration over a structure: for item in myList: / for char in myString:
- if, elif, else, for and while lines end with a colon; the block under them is indented
- Running total: total = 0 before the loop, add inside it, output after it

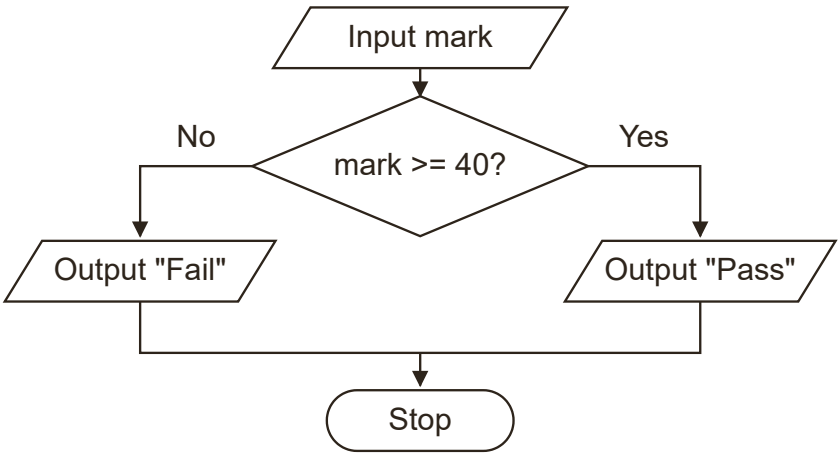
Notes

Sequence

- **Sequence**: statements run one after another, in the order they are written. Swapping two lines can change the result.
- **Initialisation** gives a variable its first value before it is used (`total = 0`); **assignment** changes its value (`total = total + price`).
- Programs are built from variables and constants, assignments, input and output, selection, repetition, iteration, data structures and subprograms.

Selection

- `if` runs its block only when its condition is `True`; the `else` block runs when it is `False`.

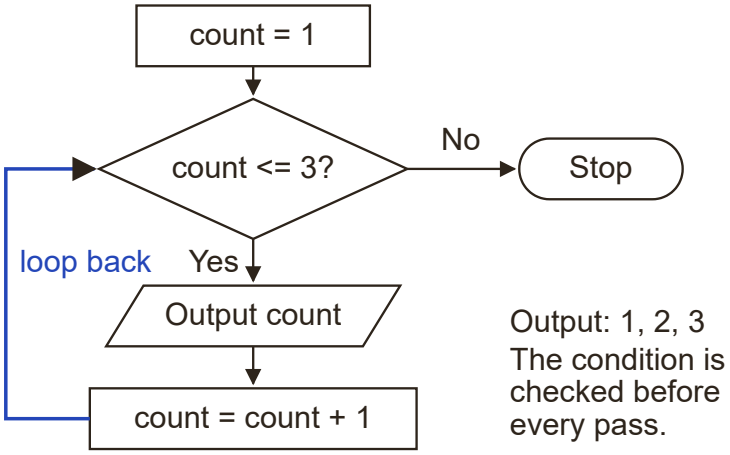


if … else as a flowchart: the condition picks one branch, then both paths join again.

- `elif` adds more conditions. They are checked from the top and only the first `True` branch runs, so when you use `>=`, test the highest value first.
- Every `if`, `elif` and `else` line ends with a colon, and the block under it is indented.
- **Nested selection** puts one `if` inside another, so the inner condition is only tested when the outer condition is `True`. Grade 7+ only

Repetition

- **Count-controlled** loop: repeats a fixed number of times. `for i in range(4):` runs 4 times, with `i` taking the values 0, 1, 2, 3.
- `range(start, stop, step)` stops **before** the stop value: `range(2, 11, 3)` gives 2, 5, 8 and `range(5, 0, -1)` gives 5, 4, 3, 2, 1.
- **Condition-controlled** loop: `while condition:` repeats while the condition is `True`. The condition is checked before each pass, so the body may never run.



A while loop as a flowchart: `while count <= 3:` repeats the two steps inside the loop until the condition is False.

- Something inside a `while` loop must eventually make the condition `False`, or the loop never ends (an infinite loop).
- Use `while` for validation, menus and 'keep going until the user enters 0'.

Iteration over a data structure

- `for item in prices:` visits every item in a list, in order; `for letter in word:` visits every character in a string.
- Use `for i in range(len(names)):` when you need the index as well, e.g. to match `names[i]` with `scores[i]`.
- **Nested loops**: the inner loop runs all the way through for every pass of the outer loop, so 3 outer passes × 4 inner passes = 12 runs of the inner block. Grade 7+ only
- Give each loop a single entry and exit point: end a `while` loop through its condition, e.g. with a Boolean flag such as `found`, rather than jumping out of it. Grade 8+ only

How to answer each type of question

State the type of loop and explain why

2 marks **Grade 5**

1. Decide whether the number of repeats is known before the loop starts.
2. Known: count-controlled (for). Not known, repeats until something happens: condition-controlled (while).
3. Give the reason using the context of the question.

Example. A program keeps asking the user for a password until the correct password is entered. State the type of loop that should be used. Give a reason for your answer. (2)

Model answer. A condition-controlled (while) loop (1), because the number of attempts the user needs is not known in advance: the loop must repeat until the correct password is entered (1).

State the output of a program with a loop

2 to 4 marks **Grade 5**

1. Write down the values the loop variable takes, e.g. from range().
2. Draw a trace table with a column for each variable and one for the output.
3. Follow the code line by line for every pass and write each output exactly, on its own line.

Example.

```
01 total = 0
02 for n in range(3, 12, 4):
03     total = total + n
04     print(n, total)
```

(a) State the output of the program. (2)
(b) State how many times line 03 is run. (1)

Model answer.

(a) `3 3`, then `7 10`, then `11 21`, each on its own line: correct values of `n` (3, 7, 11) (1); correct totals (3, 10, 21) (1)

(b) 3 (1)

Write selection code

3 to 4 marks | Grade 5

1. List the cases and their limits from the question.
2. Use if for the first case, elif for each middle case and else for the last one.
3. Check the boundaries: test values on each side of every limit (e.g. 11 and 12).

Example. A cinema charges £5 for children under 12, £9 for people aged 12 to 64 and £6 for people aged 65 or over. The age is stored in the integer variable `age`.
Write Python code that sets the variable `price` to the correct ticket price. (4)

Model answer.

```
if age < 12:
    price = 5
elif age < 65:
    price = 9
else:
    price = 6
```

Correct condition for under 12 (1); `elif` with a correct condition for 12 to 64 (1); `else` (or `age >= 65`) for 65 or over (1); correct prices with correct colons and indentation (1).

Write a program that uses a while loop

4 to 6 marks | Grade 6

1. Initialise counters and totals before the loop.
2. Input the first value before the loop, so the condition can be tested.
3. Inside the loop, process the value, then input the next one.
4. Output the results after the loop, not inside it.

Example. Write a program that asks the user to enter whole numbers until they enter 0. It must then output how many numbers were entered (not counting the 0) and their total. (6)

Model answer.

```
count = 0
total = 0
number = int(input("Enter a number (0 to stop): "))
while number != 0:
    count = count + 1
    total = total + number
    number = int(input("Enter a number (0 to stop): "))
print("Count:", count)
print("Total:", total)
```

Sets `count` and `total` to 0 (1); inputs the first number as an integer before the loop (1); a `while` loop that stops when 0 is entered (1); adds 1 to `count` and the number to `total` inside the loop (1); inputs the next number inside the loop (1); outputs the count and total after the loop (1).

Trace nested loops

2 to 3 marks | **Grade 7**

1. Take one pass of the outer loop at a time.
2. For that pass, run the inner loop all the way through before going back to the outer loop.
3. To count runs of the inner block, add up the inner passes for each outer pass.

Example.

```
01 for row in range(1, 4):
02     line = ""
03     for col in range(row):
04         line = line + "#"
05     print(line)
```

(a) State the output of the program. (2)
(b) State how many times line 04 is run. (1)

Model answer.

(a) #, then ##, then ###, each on its own line: three lines of output (1); with 1, 2 and 3 # characters (1)

(b) 6, because 1 + 2 + 3 = 6 (1)

Don’t lose marks

- Forgetting the colon at the end of an if, elif, else, for or while line: a syntax error.
- Using range(1, 10) to count from 1 to 10. It stops at 9; use range(1, 11).
- Forgetting to input again, or to change the variable, inside a while loop, so the loop never ends.
- Indenting a print() so it runs on every pass when the output is only needed once, after the loop.
- Using separate if statements where elif is needed, so more than one branch runs.
- Setting a total to 0 inside the loop, so it only ever holds the last value.

More tips

Memory tricks

- range() never reaches its stop value: think 'up to but not including'.
- range(a, b) produces b − a values, so range(1, 11) gives 10 values: 1 to 10.
- Know how many repeats? Use for. Don't know? Use while.
- With elif, the first True condition wins, so with >= start from the top value and work down.
- If the inner loop always has the same number of passes, runs of the inner block = outer passes × inner passes.

Exam technique

- Trace with a table: one column per variable and one for output, updated line by line.
- Read whether output is needed for each item (inside the loop) or once at the end (after the loop).
- Test your selection code with a value on each side of every boundary, e.g. 11 and 12, 64 and 65.
- In the onscreen exam, keep the variable names and structure you are given and add to them, rather than rewriting.

What each grade needs

What you need to be able to do, from the first marks up to the top grade.

Grade 3 — **Recognise sequence, selection and iteration in code**
Point out the lines that run in order, the lines that make a decision and the lines that repeat.

Grade 4 — **Write if, elif and else statements**
Choose between two or more paths using conditions, colons and correct indentation.

Grade 4 — **Use range() to control a for loop**
Know that range(5) gives 0 to 4 and range(1, 6) gives 1 to 5.

Grade 5 — **Choose count- or condition-controlled loops**
Use for when the number of repeats is known and while when it depends on a condition.

Grade 5 — **Trace loops to find the output**
Use a trace table to follow each variable through every pass of a loop.

Grade 6 — **Iterate over every item of a data structure**
Use for item in a list, or for character in a string, to process each one in turn.

Grade 7 — **Use nested loops and nested selection**
Put a loop or if inside another and work out how many times the inner block runs.

Grade 8 — **Combine constructs to solve unfamiliar problems**
Decompose the task, choose the right constructs and give each loop a single exit point.

Free notes, Full packs, practice papers and mock exams at chhetriacademy.com. Written for this site; not taken from past papers. Stuck? 1-to-1 tutoring with a free 30-minute trial lesson.