Syntax, logic and runtime errors

1.2.5 · Computational thinking · revision notes

Programs can contain three types of error: syntax, logic and runtime errors. You need to recognise each type, and to find and correct logic errors in algorithms. Questions ask you to identify the type of an error, explain the difference between types, and fix the lines that cause a wrong result.

Key facts

- Syntax error: breaks the rules of the language; the program will not run
- Runtime error: happens while the program is running and makes it crash
- Logic examples: wrong operator, wrong loop range, missing brackets in a formula
- Find logic errors with test data (expected v actual results) and trace tables
- Logic error: the program runs but gives the wrong result
- Syntax examples: missing colon, misspelt keyword, unmatched brackets, wrong indentation
- Runtime examples: division by zero, index out of range, invalid type conversion

Notes

Syntax errors

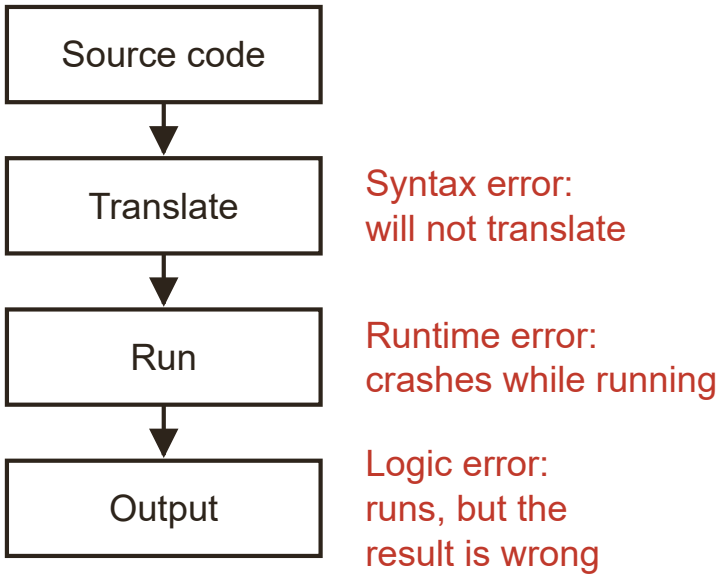
- A **syntax error** breaks the rules (grammar) of the programming language, so the code cannot be translated and the program will not run.
- Examples: a misspelt keyword (`pritn`), a missing colon after `if` or `while`, brackets or quotation marks that do not match, missing or unexpected indentation in Python.
- The translator reports syntax errors, usually with a line number, so they are the easiest type to find.

Logic errors

- A **logic error** lets the program run, but it gives the wrong result or does not do what was intended.
- There is no error message, so you find logic errors by testing with data whose correct result you already know, and by tracing.
- Common causes: the wrong relational operator (`>` instead of `>=`), `and` instead of `or`, a loop that runs one time too many or too few, missing brackets in a formula, a line indented at the wrong level (so it runs inside or outside a loop by mistake), a total that is reset inside a loop.
- Example: `average = a + b / 2` runs, but only b is divided by 2. The correction is `average = (a + b) / 2`.

Runtime errors

- A **runtime error** happens while the program is running. The syntax is correct, but the program is asked to do something impossible, so it crashes (stops).
- Examples: dividing by zero, using an index outside the array (`scores[4]` when there are only 4 items), converting text such as `"abc"` to an integer, opening a file that does not exist.



Syntax errors are found before the program runs; runtime and logic errors only appear when it runs

- Many runtime errors can be prevented by checking first: only divide if the count is not 0, check an index is less than the length of the list, and validate input before converting it. Grade 8+ only

How to answer each type of question

Identify the type of error 1 mark each **Grade 5**

1. Does the code break the rules of the language? Syntax error.
2. Does it run but give the wrong answer? Logic error.
3. Does it crash part-way through running? Runtime error.

Example. State the type of error in each of these.
(a) `print("Total" total)`
(b) `area = length + width` in a program that should calculate the area of a rectangle
(c) `average = total / count` when `count` is 0

Model answer.

(a) Syntax error (1): there is no comma between the two items

(b) Logic error (1): it should multiply, not add

(c) Runtime error (1): division by zero

Explain the difference between two types of error

2 marks | Grade 5

1. Describe the first type: what happens and when.
2. Describe the second type in the same way, so that the difference is clear.

Example. Explain the difference between a syntax error and a logic error.

Model answer. A syntax error breaks the rules of the programming language, so the program will not run (1). A program with a logic error does run, but it produces the wrong result (1).

Find and correct logic errors

2 to 4 marks | Grade 7

1. Work out the result the program should give for the data.
2. Trace the program to see what it actually does and where it goes wrong.
3. For each error, give the line number, say what is wrong and write the corrected line.

Example. This program should count how many marks are 40 or more. For this list it should output 3.

```
01 marks = [62, 40, 35, 18, 55]
02 passes = 0
03 for i in range(1, len(marks)):
04     if marks[i] > 40:
05         passes = passes + 1
06 print(passes)
```

There are two logic errors. Give the line number of each error and the corrected code.

Model answer.

Line 03 (1): the loop starts at index 1, so the first mark (62) is never checked. Correct it to `for i in range(len(marks)):` (1).

Line 04 (1): `>` does not count a mark of exactly 40. Correct it to `if marks[i] >= 40:` (1).

As written, the program outputs 1.

Don’t lose marks

- Calling a crash (such as division by zero) a logic error. It is a runtime error.
- Saying a logic error stops the program. It does not: the program runs but gives the wrong result.
- Describing a fix vaguely ('change the loop'). Write the corrected line of code.
- Changing a line that was already correct, or making a fix that introduces a new error.

More tips

Memory tricks

- Syntax: won't run. Logic: runs, but wrong. Runtime: runs, then crashes.
- A syntax error is like a spelling or grammar mistake; a logic error is like following a recipe with the wrong amount of sugar.
- Check the usual suspects for logic errors: < v <=, and v or, where a loop starts and ends, brackets, and where a total is set to 0.

Exam technique

- For each error, give three things: the line number, what is wrong, and the corrected code.
- Use the expected output given in the question: trace the code, then check that your corrected version gives that output.
- If a question says there are two errors, look for two different lines.

What each grade needs

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

Grade 3 — **Name the three types of error**
Syntax, logic and runtime errors.

Grade 4 — **Describe each type of error**
Syntax breaks the language rules; logic gives wrong results; runtime crashes while running.

Grade 5 — **Identify the type of a given error**
Decide whether it stops the program running, gives a wrong result or makes it crash.

Grade 6 — **Find a logic error using test data**
Compare the expected result with the actual result, then trace to find the line.

Grade 7 — **Correct logic errors in an algorithm**
Give the line number, what is wrong and the corrected code.

Grade 8 — **Explain how to prevent runtime errors**
E.g. check that a divisor is not zero, or that an index is in range, before using it.

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.