Solutions

Each answer shows the steps, then the **result**.

2.1

- a compiler translates the **whole** program **once** into an executable
- an interpreter translates and runs it **one line at a time**

2.2

- e.g. during development (quick edit–run cycles) / for a cross-platform script / a small or run-once program

2.3

- **assembly language** (into machine code)

2.4

- any two of: syntax highlighting; auto-complete; one-key compile/run; debugger; version-control integration

3.1

(a) any two: runs faster (no translation at run time); produces a stand-alone executable; the translator is not needed to run it; the source code is not given away

(b) any two: reports errors as it reaches them (easier development); no need to recompile after each change; the same program runs on any platform with the interpreter

3.2

- the program is compiled once into **bytecode**, which is platform-independent
- any computer with a **JVM** can interpret/run that bytecode
- so it runs unchanged on different operating systems — "write once, run anywhere"

3.3

- any two: set **breakpoints** to pause execution; **step through** line by line; **inspect variable** values while paused

3.4

- a **compiler**
- it produces a stand-alone executable that runs without any programming tools installed on the customer's computer

3.5

(a) during development an **interpreter** translates and runs one statement at a time, so the programmer sees the effect of a change immediately and can stop at the first error

- once the program works, a **compiler** translates the whole source into machine code once
- that produces an executable that runs faster and can be distributed without the source

(b) any two: an editor with syntax highlighting and auto-complete; a debugger with breakpoints, single-stepping and variable watches; a project manager for multiple source files; built-in help and error reporting

(c) free software gives the user the source code and the right to **study**, **modify** and **redistribute** it, including modified versions

- freeware costs nothing to use but the source is not provided and the licence forbids changing or redistributing it

(d) an interpreter must **translate each statement every time it is executed**, so a statement inside a loop is translated on every pass

- compiled code was translated once, before running, so the processor executes machine code directly with no translation overhead at all

(e) the real error is on an **earlier** line — for example a missing closing bracket or quotation mark — and the compiler only notices when the statement fails to make sense