Variables, constants, input, output and assignment

Algorithms and programming (Paper 2) · revision notes

Variables and constants store data in a program; assignment gives them values; input and output let the program talk to the user. These ideas appear in almost every code question, and you may be asked to define the terms, spot them in code or write short programs that use them.

Key facts

- Variable: named memory location whose value can change while the program runs
- Constant: named value that cannot change while the program runs (const in ERL)
- Assignment: variable = expression (the right side is worked out first)
- = assigns a value; == compares two values
- input() always returns a string: cast with int() or float() for numbers
- print() outputs to the screen; use str() to join numbers to text
- Use meaningful identifiers, such as totalCost, not x

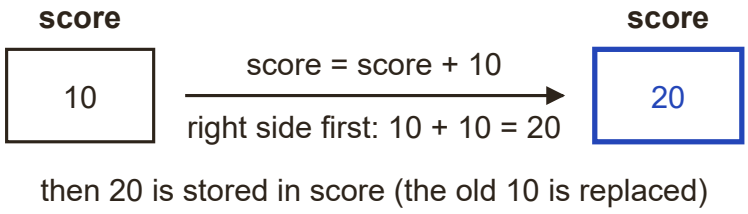
Notes

Variables and constants

- A **variable** is a named location in memory that stores a value. The value can change while the program is running.
- A **constant** is a named value that cannot change while the program is running. In ERL it is declared with `const`, for example `const VAT = 0.2`.
- Why use a constant: its value cannot be changed by mistake while the program runs, and if the value ever needs updating (for example a new VAT rate) the programmer changes it in just one place in the code.
- Use meaningful names (identifiers), such as `totalCost` rather than `x`, so the code is easy to understand.

Assignment

- **Assignment** gives a variable a value using `=`, for example `score = 0`.
- The right-hand side is worked out first, then the result is stored in the variable on the left. So `score = score + 10` adds 10 to the current score.



then 20 is stored in score (the old 10 is replaced)

Assignment: work out the right-hand side, then store it in the variable on the left

- Do not confuse `=` (assignment) with `==` (comparing two values).

Input and output

- **Input**: `name = input("Enter your name")` shows the message and stores what the user types.
- Input is always a string, so cast it to use it as a number: `age = int(input("Enter your age"))`.
- **Output**: `print("Hello " + name)` displays text on the screen. Use `str()` to join a number to a string.
- Example: a simple program.

```
const PRICE = 4.5
tickets = int(input("How many tickets?"))
cost = tickets * PRICE
print("Total: £" + str(cost))
```

How to answer each type of question

Explain why a constant is used 2 marks **Grade 4**

1. Say that the value stays the same while the program runs, so it cannot be changed by accident.
2. Say that it only needs changing in one place in the code if it is ever updated.

Example. A program uses the value 0.2 for VAT in several places. Explain why VAT should be declared as a constant rather than a variable. [2]

Model answer. The value does not need to change while the program runs, so declaring it as a constant stops it being changed by accident (1). If the VAT rate changes, it only needs to be updated in one place in the code (1).

State the values after a sequence of assignments

2 to 3 marks | Grade 5

1. Work through one line at a time, keeping a note of each variable's current value.
2. Always use the value a variable has at that moment, not its starting value.
3. If asked what the code does, describe the overall effect.

Example. State the final values of `a` and `b`, and describe what the code does.

```
a = 4
b = 7
a = a + b
b = a - b
a = a - b
```

[3]

Model answer. a = 7 (1), b = 4 (1). It swaps the values of a and b (1). (a becomes 11, then b = 11 − 7 = 4, then a = 11 − 4 = 7.)

Write a short input, process, output program

3 marks | Grade 6

1. Input the data and cast it to the right type.
2. Do the calculation and store the result in a variable.
3. Output the result.

Example. Write a program that asks the user for a temperature in degrees Celsius and outputs it in degrees Fahrenheit, using F = C × 1.8 + 32. [3]

Model answer.

```
c = float(input("Enter temperature in Celsius"))
f = c * 1.8 + 32
print(f)
```

Input stored in a variable and cast to a real number (1); correct calculation stored in a variable (1); outputs the result (1).

Don’t lose marks

- Saying a constant 'can never be changed': the programmer can edit it in the code; it just cannot change while the program runs.
- Writing assignments backwards, such as `10 = score`.
- Doing arithmetic on input without casting it: "5" + "3" gives "53", not 8.
- Using = when you mean == in a condition.
- Joining a number to a string without str().

More tips

Memory tricks

- Constant = Can't change (while running). Variable = Varies.
- Read assignment from right to left: 'work this out, then put it in there'.
- When you see input() in a calculation, check for int() or float() around it.

Exam technique

- Definitions of variable and constant should mention 'while the program is running'.
- If a question gives variable names, use exactly those names in your code.
- Initialise variables (for example total = 0) before using them in a calculation.

What each grade needs

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

Grade 3 — **Define a variable and a constant**
A variable's value can change while the program runs; a constant's value cannot.

Grade 4 — **Identify variables, constants and assignments in code**
Point to the lines that declare a constant, assign a value, take input or give output.

Grade 4 — **Explain why a constant is used**
It cannot be changed by mistake, and it only needs updating in one place in the code.

Grade 5 — **Trace a sequence of assignments**
Work out each variable's value line by line, remembering the right side is worked out first.

Grade 6 — **Write code that inputs, processes and outputs**
Take input, cast it to the right type, calculate a result and output 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.