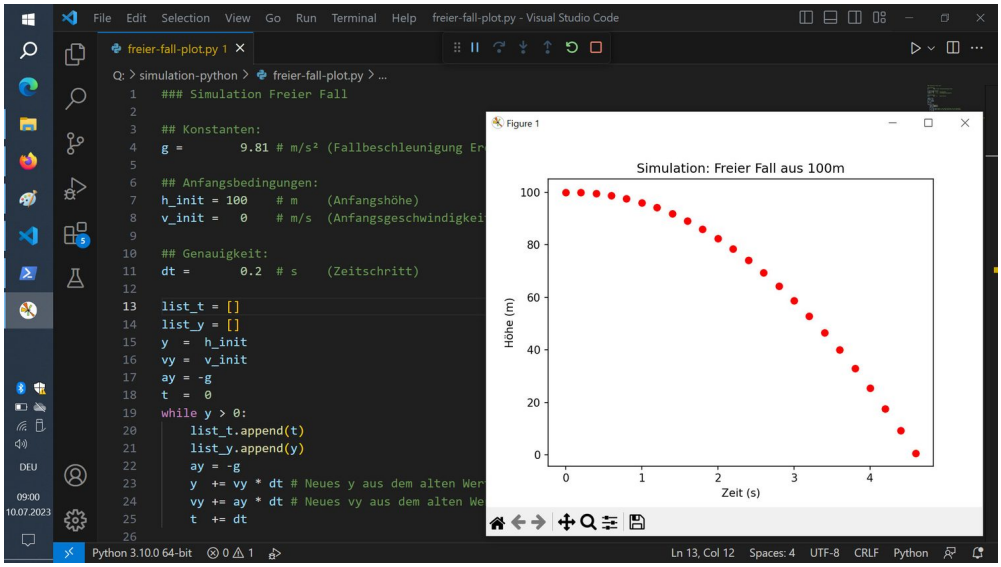
Programming

IGCSE Computer Science

Variables and constants

A **variable** 变量 is a named store that holds a value which **can change** while the program runs. A **constant** 常量 is a named store whose value is **fixed** and does not change.



Variables and constants are named stores for values, set in the program's code

Image: MikeRun, CC BY-SA 4.0 (commons.wikimedia.org)

You should **declare** 声明 them (say their name and type) before use:

```
DECLARE score : INTEGER
DECLARE name : STRING
CONSTANT Pi = 3.142
```

While a program runs, its variables are held in the computer's memory (RAM)

Image: Veeblefetzer, CC BY 4.0 (commons.wikimedia.org)

Use a constant for a value that never changes (like Pi), so it is set in one place and is easy to read.

Data types

A **data type** 数据类型 says what kind of value a variable holds. You must know five basic types.

Type	Holds	Example
integer 整数	a whole number	`42`, `-7`
real 实数	a number with a decimal point	`3.14`, `-0.5`
char 字符	a single character	`'A'`, `'?'`
string 字符串	a sequence of characters	`"Hello"`
Boolean 布尔值	one of two values	`TRUE` or `FALSE`

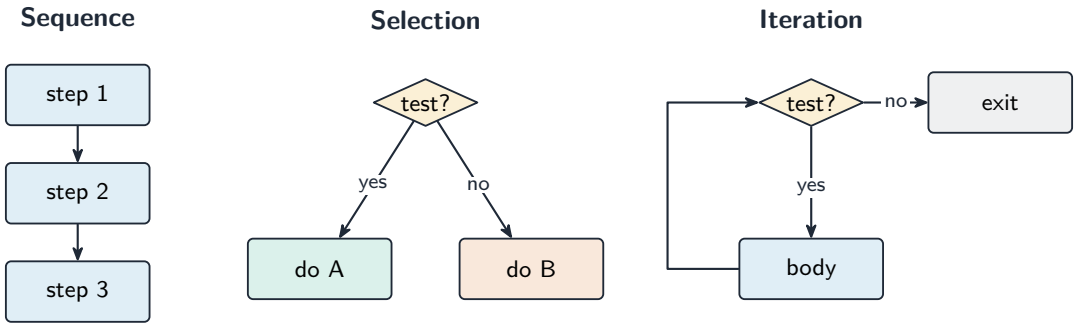
Input and output

Input 输入 reads a value from the user. **Output** 输出 shows a value on the screen.

```
OUTPUT "What is your name?"
INPUT name
OUTPUT "Hello ", name
```

The three basic structures

Every program is built from three control structures.



The three control structures: sequence runs steps in order, selection chooses a branch, iteration repeats a body

Sequence

Sequence 顺序 means the steps run one after another, in order, from top to bottom.

```
INPUT length
INPUT width
area ← length * width
OUTPUT area
```

Selection

Selection 选择 chooses which steps to run, based on a condition. Use an `IF` statement, or a `CASE` statement when there are many choices.

```
IF score >= 50 THEN
    OUTPUT "Pass"
ELSE
    OUTPUT "Fail"
ENDIF
```

```
CASE OF grade
    'A' : OUTPUT "Excellent"
    'B' : OUTPUT "Good"
    OTHERWISE : OUTPUT "Keep trying"
ENDCASE
```

Iteration

Iteration 迭代 (a loop 循环) repeats steps. There are three kinds.

A **count-controlled loop** 计数循环 repeats a fixed number of times:

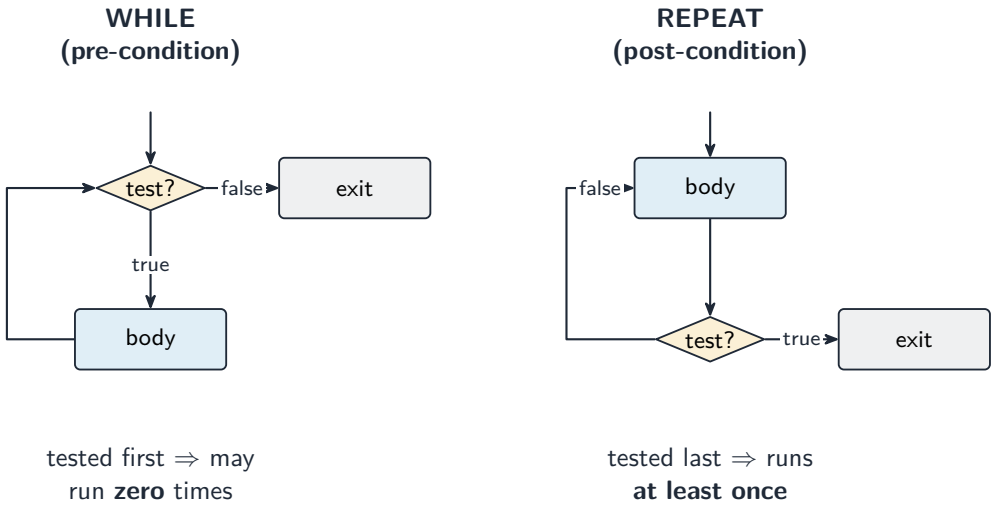
```
FOR i ← 1 TO 5
    OUTPUT "Hello"
NEXT i
```

A **pre-condition loop** 前测循环 checks the condition **before** each repeat, so it may run zero times:

```
WHILE answer <> "stop" DO
    INPUT answer
ENDWHILE
```

A **post-condition loop** 后测循环 checks the condition **after** each repeat, so it always runs **at least once**:

```
REPEAT
    INPUT password
UNTIL password = "secret"
```



A pre-condition (WHILE) loop tests before the body, so it may run zero times; a post-condition (REPEAT) loop tests after, so it runs at least once

Totalling and counting

- **totalling** 求和—keep adding values to a running total: `total ← total + value`.
- **counting** 计数—add 1 to a counter each time: `count ← count + 1`.

```
total ← 0
FOR i ← 1 TO 10
    INPUT mark
    total ← total + mark
NEXT i
OUTPUT total
```

Operators

Arithmetic operators

Operator	Meaning
`+ - * /`	add, subtract, multiply, divide
`^`	raised to the power of
`MOD`	the **remainder** 余数 after division
`DIV`	the whole-number part of a division

For example, `17 MOD 5` is `2`, and `17 DIV 5` is `3`.

Relational operators

These compare two values and give a Boolean result: `=`, `<`, `<=`, `>`, `>=`, and `<>` (not equal to).

Logical operators

These join conditions: `AND` (both must be true), `OR` (at least one must be true), `NOT` (reverses true/false).

```
IF age >= 13 AND age <= 19 THEN
    OUTPUT "Teenager"
ENDIF
```

String handling

A string is made of characters. Useful operations:

- **length** 长度—the number of characters in the string;
- **substring** 子串—a smaller part taken from the string;
- **upper case** 大写—change letters to capitals;
- **lower case** 小写—change letters to small letters.

```
name ← "Computer"
OUTPUT LENGTH(name)          // 8
OUTPUT SUBSTRING(name, 1, 4) // "Comp"
OUTPUT UCASE(name)           // "COMPUTER"
OUTPUT LCASE(name)           // "computer"
```

(The first character may be counted as position 0 or position 1, depending on the language.)

Nested statements

A **nested statement** 嵌套语句 is one control structure placed **inside** another. You can nest selection and iteration.

```
FOR i ← 1 TO 3
    IF i MOD 2 = 0 THEN
        OUTPUT i, " is even"
    ELSE
        OUTPUT i, " is odd"
    ENDIF
NEXT i
```

You will not have to write more than three levels of nesting.

Procedures and functions

To avoid repeating code, you can break a program into named blocks.

- A **procedure** 过程 is a named block of code that does a task. You run it with `CALL`.
- A **function** 函数 is like a procedure, but it **returns a value** back to where it was called.

A **parameter** 参数 is a value passed into a procedure or function (up to three parameters).

```
PROCEDURE Greet(personName : STRING)
    OUTPUT "Hello ", personName
ENDPROCEDURE

CALL Greet("Sam")
```

```
FUNCTION Square(n : INTEGER) RETURNS INTEGER
    RETURN n * n
ENDFUNCTION

answer ← Square(5)   // answer is 25
```

Local and global variables

- A **local variable** 局部变量 is declared inside a procedure or function. It can only be used there.
- A **global variable** 全局变量 is declared in the main program. It can be used anywhere.

Local variables are safer, because they cannot be changed by mistake from another part of the program.

Library routines

A **library routine** 库程序 is a ready-made piece of code you can use. You must know these:

Routine	What it does
`MOD`	gives the remainder of a division
`DIV`	gives the whole-number part of a division
`ROUND`	rounds a real number to a number of decimal places
`RANDOM`	gives a random number

Writing a maintainable program

A **maintainable** 可维护的 program is easy for other people to read and change later. To make one:

- use **meaningful identifiers** 有意义的标识符—clear names for variables, constants, arrays, procedures and functions (`totalScore`, not `x`);
- add **comments** 注释 to explain what parts of the code do;
- use procedures and functions to split the work into small blocks.

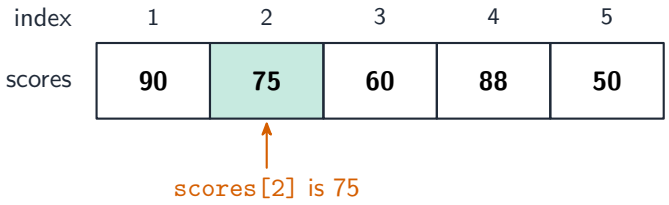
Arrays

An **array** 数组 is a single variable that holds **many values** of the same type, found by an **index** 索引 (a position number).

One-dimensional arrays

A **one-dimensional (1D) array** 一维数组 is like a single list.

```
DECLARE scores : ARRAY[1:5] OF INTEGER
scores[1] ← 90
scores[2] ← 75
OUTPUT scores[1]
```



A 1D array holds many values in one variable; each is found by its index

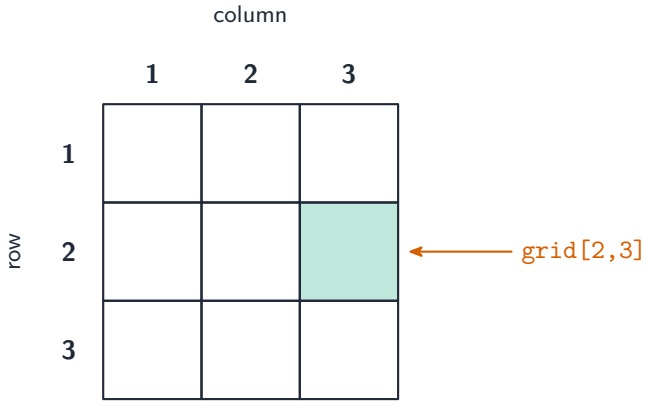
You can use a loop to fill or read an array:

```
FOR i ← 1 TO 5
    INPUT scores[i]
NEXT i
```

Two-dimensional arrays

A **two-dimensional (2D) array** 二维数组 is like a table with rows and columns. It uses two indexes.

```
DECLARE grid : ARRAY[1:3, 1:3] OF INTEGER
grid[1, 1] ← 5
grid[2, 3] ← 8
```



A 2D array is a table; each value is found by two indexes, [row, column]

You read a 2D array with a loop inside a loop (nested iteration).

File handling

A program can store data in a **file** 文件 so it is kept after the program stops. You must **open** the file, use it, then **close** it.

```
OPENFILE "data.txt" FOR WRITE
WRITEFILE "data.txt", "Hello"
CLOSEFILE "data.txt"

OPENFILE "data.txt" FOR READ
READFILE "data.txt", line
CLOSEFILE "data.txt"
```

You can read and write single items of data or a whole line of text. Always close a file when you have finished with it.