Variables, constants and data structures (strings, arrays and records)

1.2.2 · Computational thinking · revision notes

Programs store data in variables and constants, and group data into data structures: strings, one- and two-dimensional arrays, and records. Questions ask you to define these terms, state the value stored at a given index, explain why a constant is used, and write code that works with arrays and records.

Key facts

- Variable: a named memory location whose value can change while the program runs
- Constant: a named value that stays the same while the program runs (Python convention: CAPITALS)
- String: a sequence of characters; the first character has index 0
- len() gives the number of characters in a string or items in a list
- Array: many items of the same data type under one identifier, accessed by an index from 0
- 2D array: rows and columns, table[row][column]
- Record: related fields about one item; the fields can be different data types
- Last index = length − 1

Notes

Variables and constants

- A **variable** is a named location in memory that holds a value that can change while the program runs.
- In `score = score + 10` the right-hand side is worked out first, then the result is stored in `score`.
- A **constant** is a named value that does not change while the program runs. Python has no true constants, so by convention they are written in capitals and never reassigned, e.g. `VAT_RATE = 0.2`.
- Benefits of a constant: if the value ever needs updating it is changed in one place only, and the name is easier to understand than a bare number. In languages that enforce constants, the value also cannot be changed by mistake while the program runs.
- Use meaningful identifiers, such as `totalCost` rather than `x`.

Strings

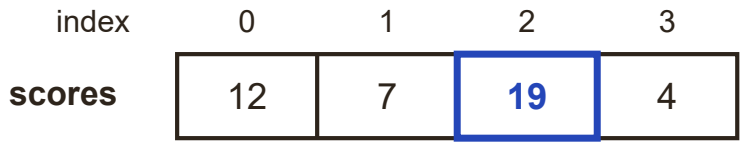
- A **string** is a sequence of characters in quotation marks. Each character has an index, starting at 0.
- This outputs P, then 6, then PYTHON3.

```
word = "PYTHON"
print(word[0])
print(len(word))
print(word + "3")
```

- To join a number to a string, convert it first: `"Score: " + str(score)`.

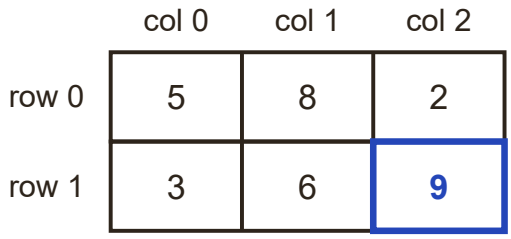
Arrays

- An **array** stores many items of the same type under one identifier. Each item is found by its index, starting at 0. In Python, arrays are written as lists.
- One-dimensional: in `scores = [12, 7, 19, 4]`, `scores[2]` is 19 and `scores[1] = 8` changes the second item. The last index is `len(scores) - 1`.



scores[2] is 19

Each item has an index, starting at 0. The last index is len(scores) − 1 = 3



sales[1][2] is 9: row first, then column

A 2D array is a table: the first index picks the row, the second picks the column

- Two-dimensional: a table of rows and columns, written as a list of lists. The first index picks the row and the second picks the column. This outputs 9 (row 1, column 2).

```
sales = [[5, 8, 2],
         [3, 6, 9]]
print(sales[1][2])
```

Records

- A **record** groups the related data items (**fields**) about one thing. The fields can be different data types.
- In Python a record can be stored as a list: in `pupil = ["P104", "Amira", 15]`, `pupil[1]` is the name field.
- A table of records is a two-dimensional structure (a list of records): `pupils[0][1]` is the name of the first pupil.
- Array or record? An array holds many items of the same type (e.g. 30 test scores); a record keeps different types of data about one item together.

How to answer each type of question

State the difference between a variable and a constant 1 mark **Grade 3**

1. Say what happens to the value while the program is running.
2. Variable: the value can change. Constant: the value stays the same.

Example. State **one** difference between a variable and a constant.

Model answer. The value of a variable can change while the program is running, but the value of a constant cannot (1).

State the value stored in a data structure 1 mark each **Grade 5**

1. Count indexes from 0.
2. For a two-dimensional array, find the row first, then the column.
3. Give the value exactly as it is stored.

Example. Here is a two-dimensional list.

```
stock = [["pen", 40, 0.5],
         ["ruler", 12, 1.2],
         ["glue", 5, 2.75]]
```

State the value of:
(a) `stock[1][0]`
(b) `stock[2][1]`
(c) `len(stock)`

Model answer.

(a) ruler (1)
(b) 5 (1)
(c) 3 (1), because there are three records

Explain why a constant is used

2 marks **Grade 5**

1. Give a benefit: the value is set in one place only, or the name explains what the value means.
2. Add the consequence, linked to the program in the question.

Example. A program for a shop uses the constant `VAT_RATE = 0.2` in several calculations. Explain **one** benefit of using a constant for the VAT rate.

Model answer.

If the VAT rate changes, the programmer only needs to change the value in one place (1), so every calculation that uses it is updated and none are missed (1).

Or: the name VAT_RATE shows what the number means (1), so the code is easier to read and maintain (1).

Choose a suitable data structure

2 marks **Grade 7**

1. Say what the data is like: one item or many, the same type or mixed types.
2. Link this to a feature of the structure: a record holds fields of different types about one item; an array holds many items of one type.

Example. A program stores the title, the year of release and the star rating (such as 4.5) of one film. Explain why a record is a suitable data structure for this data.

Model answer. The data items are all about one film, so a record keeps them together (1). They are different data types (a string, an integer and a real number), and the fields of a record can be different types (1).

Write code that processes a list of records

3 to 6 marks **Grade 8**

1. Loop through every record in the list.
2. Pick out the field you need using its index.
3. Apply the condition or calculation, then output the correct field.

Example. Here is a two-dimensional list. Each record holds an item's name, the number in stock and the price.

```
stock = [["pen", 40, 0.5],
         ["ruler", 12, 1.2],
         ["glue", 5, 2.75]]
```

Write Python code that outputs the name of every item that has fewer than 20 in stock.

Model answer.

```
for item in stock:
    if item[1] < 20:
        print(item[0])
```

Loops through every record (1); uses the number in stock, index 1 (1); a correct comparison, less than 20 (1); outputs the name field, index 0 (1).

For this list the output is ruler, then glue.

Don’t lose marks

- Counting indexes from 1 instead of 0.
- Swapping the row and the column in a two-dimensional array.
- Saying a constant can never be changed. The programmer can edit it in the code; it just does not change while the program runs.
- Joining a number to a string with `+` without using `str()` first.
- Using an index equal to the length of the list, e.g. `scores[4]` for a list of 4 items. This causes an 'index out of range' runtime error.

More tips

Memory tricks

- Row first, then column: `grid[row][column]`. Go down to the row, then across.
- Indexes start at 0: the 5th item is `[4]` and the last item is at length − 1.
- CAPITALS say 'constant: do not change me'.
- Record = one row of a table (one thing).
  Field = one column (one piece of data about it).

Exam technique

- When you state a value from a two-dimensional array, check it by saying 'row ..., column ...' to yourself.
- In 'explain' questions about constants, link your answer to the value in the question, such as a tax rate or a maximum score.
- When writing code, use exactly the identifiers and field positions given in the question.

What each grade needs

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

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

Grade 4 — **Access an array item by its index**
Indexes start at 0, so `scores[2]` is the third item.

Grade 5 — **Explain the benefits of using constants**
The value is changed in one place only, and the name makes the code easier to read.

Grade 5 — **Use string length, indexing and concatenation**
Use `len()`, `word[0]` and `+` to work with strings.

Grade 6 — **Use a two-dimensional array**
Find or change an item using a row index and then a column index.

Grade 7 — **Choose between an array and a record**
A record holds related fields of different types; an array holds items of the same type.

Grade 8 — **Process arrays and records with loops**
Loop through every item or record to total, count, search or output fields.

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.