



Arrays, Strings and Records

GCSE student booster

Date

Key Information

- 1) Remember this booster is here to **help you**. Please consider your behaviour in the chat.
- 2) If you are in a room with a teacher/group, please login to the meeting. This is so we can mark your attendance. This information goes into a **prize draw**.
- 3) Make sure the name on the meeting is the **SAME** as the name on your Isaac account. We can't mark you present if they don't match.



Starter

2 minutes

- What are the similarities and differences between a variable and a data structure?

```
• my_variable = 1
• my_data_structure = [1,2,3,4]
```

- Answers in the chat



Answers

Similarities

- Both have identifiers (names).
- Both have a memory location in RAM.

Differences

- A variable holds a single piece of data while a data structure **can** hold multiples pieces of data.



What was the point of the starter?

- This question reviews how data is stored before learning arrays.
- It prepares you for working with collections of data.

Arrays – Ordered collections of data of the same type.

Strings – Sequences of characters treated as text.

Records – Structured data with fields of different types



Intended learning outcomes

- Be able to explain how **one-dimensional and two-dimensional arrays** are used to organise and manipulate data.
- Be able to implement and work with **arrays and records** to store and process structured data.
- Be able to perform key **string operations** such as concatenation, slicing, and searching, and integrate them with arrays.
- Be able to compare the use of **arrays, records, and strings** for solving computational problems.



Check in

1 minute

On a scale of 1–10 (1 = low, 10 = high), rate your confidence in:

1. Understanding what a data structure is.
2. Using arrays/lists to solve simple problems.
3. Explaining the difference between lists and arrays.

Post your answers in the chat, e.g., 1(5), 2(7), 3(4)



Data Structures

- A data structure is any method used to store data in an organised and accessible format
- Different data structures lend themselves to different applications
- An array/list might be used to temporarily store the data while the program is running
- A text file might be used to store the data for further use



Lists vs Arrays

Feature	Python Lists	Arrays
Data Type	Can hold elements of different data types.	Typically holds elements of a single data type.
Size	Can grow or shrink as needed.	Fixed size; defined at the time of creation.
Operations	Supports various built-in methods like <code>append()</code> , <code>insert()</code> , <code>del()</code> and more.	Basic functionality, mainly focused on accessing and updating elements using an index.
Usage	Suitable for applications where the number of elements is not known in advance and can change.	Arrays are generally more efficient for numerical computation as they can handle large data sets.



Python Lists

```
names = ["Thiago", "Fatima", "James"]
```

- Each item is separated from the next item by a comma.
- If the data is a string, then it will sit inside quote marks.
- The whole list sits inside a pair of square brackets.



List Basics

- Lists can contain mixed data types

```
student = ["Emma", "Johnson", 18]
```

- Access elements using their index position

```
print(student[0])
```

```
> Emma
```

- Retrieve multiple elements with list[start:end]

```
print(student[0:2])
```

```
> ["Emma", "Johnson"]
```



List Features

- List items can be changed

```
names[0] = "Emma"
```

```
print(names)
```

```
> ["Emma", "Fatima", "James"]
```

- You can loop through lists with a for loop

```
for index in range(0, len(names)-1):
```

```
    print(names[index])
```

```
> Emma
```

```
> Fatima
```

```
> James
```



List Operations

- Use list-specific operations like `append()`

```
names.append("Ruth")
print(names)
> ["Emma", "Fatima", "James", "Ruth"]
```
- Use `+` to combine two lists

```
[1, 2] + [3, 4]
> [1, 2, 3, 4]
```
- Use `len()` to find the number of items in a list

```
len(names)
> 4
```



Predict and Run


[Trinket Link](#)


Activity 1

```

1 # Predict and Run
2 student = ["Emma", "Johnson", 18]
3 names = ["Thiago", "Fatima", "James"]
4
5 # Q1
6 print(student[1:2])
7 print(student[2])
8
9 # Q2
10 try:
11     names[3] = "Radio"
12     print(names)
13 except Exception as error:
14     print(error)
15
16 # Q3
17 for name in names:
18     print("Hello, " + name)
19
20 # Q4
21 names.append("Ruchi")
22 names.append("Rana")
23 print(names)
24
25 # Q5
26 extra_names = ["Marissa", "Fei-fei"]
27 names = names + extra_names
28 print(names)
29 print(len(names))

```

Powered by trinket

```

["Johnson", 18]
["Emma", "Johnson"]
list assignment index out of range
Hello, Thiago
Hello, Fatima
Hello, James
["Thiago", "Fatima", "James", "Ruchi", "Rana"]
["Thiago", "Fatima", "James", "Ruchi", "Rana", "Marissa", "Fei-fei"]

```



Game Board


[Isaac link](#)


Gameboard

Home > Arrays Gameboard

Arrays Gameboard

How many students?
Data structures and algorithms > Data structures

Arrays
Data structures and algorithms > Data structures

Index numbers
Data structures and algorithms > Data structures

Save to My gameboards



Characteristics of a string

- A string is a sequence of characters (letters, numbers, symbols, spaces) enclosed in quotes.
- Strings can store alphanumeric data, including leading zeros (e.g. "0123").
- Strings are important for formatted numbers e.g. Phone numbers.

```
phone_no = "0123456789"
print(phone_no)
> 0123456789

phone_no = 0123456789
print(phone_no)
> Syntax Error

print(str(phone_no))
> 123456789
```



String Length

- A string's length is the number of characters it contains, including spaces and symbols.
- Use the built-in len() function to return the length of a string.

```
password = "secret"
length = len(password)
print(length)
> 6
```



Empty Strings

- An empty string contains no data and has a length of 0
- An empty string is different from a string containing a space

```
empty_string = ""
string_with_space = " "
print(len(empty_string)) # Output: 0
print(len(string_with_space)) # Output: 1
```



String Case

- String case affects how text is displayed upper, lower, title, sentence, or capital case.
- Functions like UPPER() and LOWER() are used to change string case in most programming languages.

```
user_name = input("Enter your full name: ")
upper_case_name = user_name.upper()
print(upper_case_name)
# Output: NAME IN UPPER CASE
```



String Concatenation

- String concatenation joins two or more strings together using operators like +.
- You can concatenate strings with other data types by converting the data type to a string first.

```
name = "Emma Johnson"
age = 18
print(name + " is " + str(age))
# Output: Emma Johnson is 18
```

Why does this code have str in?



String indexing

animal	Index	0	1	2
	String	C	A	T

- Indexing starts at 0:
- The length of "CAT" is 3 characters
- The index for character "T" is 2
- To retrieve a letter by index, use string[index]
e.g. "CAT"[2] returns "T"
- Predict: What might happen if you tried? `print (animal[1])`



Substrings

- A substring is a sequence of characters within a string that are next to each other.
- For example, "Isaa", "omp", and "er" are all substrings of "Isaac Computer Science"
- Substrings must contain characters that appear in the same order and next to each other in the original string.
- For example, "apt" is not a substring of "Isaac Computer Science"



Slicing

- Slicing allows you to extract a part (substring) of a string using index positions.
- In Python, slicing is done using the syntax `string[start:end]`
- For example, "Computer Science"[0:8] will return "Computer" because it extracts characters from index 0 up to, but not including, index 8.
- Negative indexing can be used to slice from the end of the string, e.g. `string[-7:]` extracts the last 7 characters.



Predict & Run



```

1 # Predict and Run
2
3 # Q1: String Indexing
4 name = "Isaac Newton"
5 print(name[0])
6 print(name[5])
7
8 # Q2: String Slicing
9 subject = "Computer Science"
10 print(subject[0:8])
11 print(subject[-7:])
12
13 # Q3: String Concatenation
14 first_name = "Ada"
15 last_name = "Lovelace"
16 full_name = first_name + " " + last_name
17 print(full_name)
18
19 # Q4: String Length
20 password = "secretPassword123!"
21 length = len(password)
22 print(length)
23
24 # Q5: Changing String Case
25 sentence = "I love learning Python!"
26 upper_sentence = sentence.upper()
27 lower_sentence = sentence.lower()
28 print(upper_sentence)
29 print(lower_sentence)
30
31 # Q6: Empty Strings
32 empty_string = ""
33 string_with_space = " "
34 print(len(empty_string))
35 print(len(string_with_space))

```

Powered by  trinket

```

I
N
Computer
Science
Ada Lovelace
18
I LOVE LEARNING PYTHON!
i love learning python!
0
1

```



Strings vs Lists

Feature	Strings	Lists
Type	Only contains characters	Can contain elements of mixed types
Creation	Defined with quotes (" or ")	Defined with square brackets []
Indexing	Supports indexing	Supports indexing
Slicing	Supports slicing	Supports slicing
Iteration	Can be iterated over characters	Can be iterated over elements
Functions	String-specific operations e.g. upper(), lower()	List-specific operations e.g. append()
Concatenation	Can concatenate with +	Can concatenate with +
Length Function	Length obtained with len()	Length obtained with len()



Check in



Given the following a string and a list in Python, which of the following statements is true?

```
my_string = "hello"
my_list = ['h', 'e', 'l', 'l', 'o']
```

- A. You can change an element of both the string and the list.
- B. You can add an element to both the string and the list using the append() method.
- C. You can access the first element of both the string and the list using my_string[0] and my_list[0] respectively.
- D. You can concatenate a list to a string, but not a string to a list.



Check in(Answer)

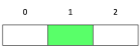
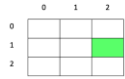


Given the following a string and a list in Python, which of the following statements is true?

```
my_string = "hello"
my_list = ['h', 'e', 'l', 'l', 'o']
```

- A. You can change an element of both the string and the list.
- B. You can add an element to both the string and the list using the append() method.
- C. You can access the first element of both the string and the list using my_string[0] and my_list[0] respectively.
- D. You can concatenate a list to a string, but not a string to a list.



Feature	One-Dimensional List	Two-Dimensional List
Abstraction:		
Indexing	Single index (e.g., <code>list[1]</code> accesses the second element).	Two indices (e.g., <code>list[1][2]</code> accesses the third element in the second row).
Structure	A single series of elements.	A list of lists, where each sub-list represents a row.
Common Uses	Storing sequences like days of the week or names.	Complex data structures like matrices, game boards, or tables.
Accessibility	Access elements directly through their position in the list.	Access elements through their position within a row and column.
Example	<code>days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri']</code>	<code>matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]</code>

2D Lists



Activity 4

- Given the two-dimensional list:

```
grades = [[88, 76, 92],
          [85, 90, 89],
          [91, 74, 88]]
```

- How would you access the grade 90? Provide the Python code necessary to retrieve this grade.

```
grades[1][1]
```

2D List Solution



Activity 4a

```

1 # 2D lists
2
3 grades = [[88, 76, 92],
4           [85, 90, 89],
5           [91, 74, 88]]
6
7 # Q1
8 first_grade_first_student = grades[0][0]
9 print(first_grade_first_student)
10 last_grade_last_student = grades[2][2]
11 print(last_grade_last_student)
12
13 # Q2
14 for student_grades in grades:
15     average = sum(student_grades) // len(student_grades)
16     print("Average grade:", average)
17
18

```

Result

Powered by trinket

```

88
88
Average grade 85
Average grade 88
Average grade 84

```

Records

- A record is a data structure that groups related data.
- Records contain fields, each storing a specific value.
- Fields can have different data types (e.g. string, integer).
- Useful for organising complex data like database entries.



Check-in

- A car rental system stores details about each car booking as a record. A typical booking record is shown in the table:

Field	Value
firstName	Lewis
surName	Green
daysRented	4
carType	SUV
insurance	True

- State the most appropriate data type for the following fields i) daysRented ii) carType



Check-in

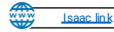
- A car rental system stores details about each car booking as a record. A typical booking record is shown in the table:

Field	Value
firstName	Lewis
surName	Green
daysRented	4
carType	SUV
insurance	True

- Give the name of one field that could be stored as a Boolean data type?



Game Board



Arrays, Strings and Record GCSE Booster		
1	Indexing	100%
2	Getting the position	100%
3	Splicing	100%
4	Board game starts	100%
5	Arrays	100%
6	How many students?	100%
7	Index numbers	100%
8	Two-dimensional array	100%
9	One- and two-dimensional arrays	100%
10	That's not correct!	100%

Intended learning outcomes

- Be able to explain how **one-dimensional** and **two-dimensional arrays** are used to organise and manipulate data.
- Be able to implement and work with **arrays** and **records** to store and process structured data.
- Be able to perform key **string operations** such as concatenation, slicing, and searching, and integrate them with arrays.
- Be able to compare the use of **arrays**, **records**, and **strings** for solving computational problems.

ISAAC boosters

Keep an eye out
for more student
booster events

Thank you