Teaching London Computing

Programming for GCSE Topic 6.1: Lists (Arrays) and For Loop

Queen Mary University of London

King's College London

Computing At School
EDUCATE · ENGAGE · ENCOURAGE

Department for Education

SUPPORTED BY
MAYOR OF LONDON

Network of Excellence Computer Science Teaching
UNIVERSITY PARTNER

William Marsh
School of Electronic Engineering and Computer Science
Queen Mary University of London

Outline

- Array in Python – the issues
- Lists – behaviour that is like an array
  - Looping through a list
- Lists – other behaviour
- For loops

Arrays in Python – The Issue

- Python does not have arrays
  - (A slight simplification)
- There are two alternatives
  - Lists
  - Dictionaries
  - Both are more flexible than ‘ordinary’ arrays
- Lists
  - Simpler
  - 'Array-like' behaviour

Big Idea

- A variable can have a value that combines many values. You can:
  - Extract one value
  - Update part of the variable
- This idea is essential for representing complex data in programs e.g.
  - A song
  - An image
  - A map

Array Behaviour of Lists

‘Simple’ Arrays

- Size (length) is fixed
  - When you start using it
- Can
  - Select (i.e. index) one entry
  - Update one entry
- Cannot
  - Add an extra entry to the start/end
  - Insert/remove item from middle

Index Into An Array

- Select an entry from the array
  - Numbered from [0]
  - … up length - 1

```
>>> myl = [9,3,5,6,4,3,2]
>>> myl[1]
3
>>> myl[0]
9
>>> myl[3] + myl[5]
9
>>>
```

Update An Entry

- Change an entry

```
>>> myl
[9, 3, 6, 5, 4, 3, 2]
>>> myl[3] = -1
>>> myl[4] = myl[4] - 2
>>> myl
[9, 3, 6, -1, 2, 3, 2]
```

- New form of assignment
  - array[number] = …

Quiz – Swap?

- What is the list after this:

```
>>> myl
[9, 3, 5, 6, 4, 3, 2]
>>> myl[2] = myl[3]
>>> myl[3] = myl[2]
>>> myl
??????
```

Quiz – Swap?

- What is the list after this:

```
>>> myl
[9, 3, 5, 6, 4, 3, 2]
>>> myl[2] = myl[3]
>>> myl[3] = myl[2]
>>> myl
[9, 3, 6, 6, 4, 3, 2]
>>>
```

Correct Swap

- Use another variable

```
>>> myl
[9, 3, 5, 6, 4, 3, 2]
>>> temp = myl[2]
>>> myl[2] = myl[3]
>>> myl[3] = temp
>>> myl
[9, 3, 6, 5, 4, 3, 2]
```

Python Problem

- Beginners are unlikely to encounter this problem but …

```
>>> myl
[9, 3, 6, -1, 2, 3, 2]
>>> myl2 = myl
>>> myl2
[9, 3, 6, -1, 2, 3, 2]
>>> myl[0] = 0
```

New copy of myl ?

Change one copy

Python Problem

- Beginners are unlikely to encounter this problem but …

```
>>> myl
[9, 3, 6, -1, 2, 3, 2]
>>> myl2 = myl
>>> myl2
[9, 3, 6, -1, 2, 3, 2]
>>> myl[0] = 0
>>> myl2
[0, 3, 6, -1, 2, 3, 2]
>>>
```

New copy of myl ?

Change one copy

Both changed

Looping Through an Array

Loop Through an Array

- Counter from 0 up to (but not including) len(myl)

```
myl = [...]
cntr = 0
while cntr < len(myl):
    print("Item", myl[cntr])
    cntr = cntr + 1
```

Lists: Beyond Arrays

Joining Lists

- Two can be concatenated

```
>>> list1
[1, 1, 1]
>>> list2
[2, 2]
>>> list1 + list2
[1, 1, 1, 2, 2]
>>>
```

Changing Length

- Append

```
>>> ones
[1, 1, 1]
>>> ones.append(2)
>>> ones
[1, 1, 1, 2]
>>>
```

- Remove

```
>>> ones.remove(2)
>>> ones
[1, 1, 1]
>>> ones.remove(2)
Traceback (most recent call last):
  File "<pyshell#62>", line 1, in <module>  ones.remove
ValueError: list.remove(x): x not in list
>>>
```

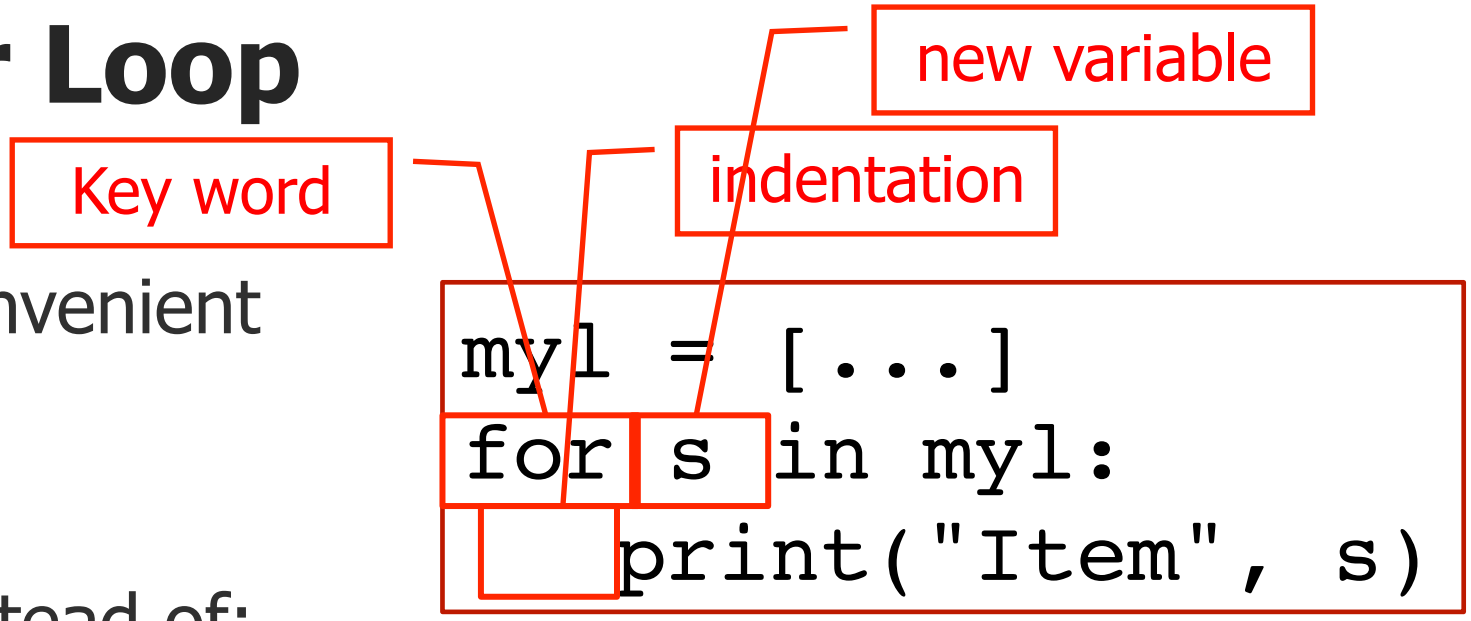
Membership Test

- Test it a value is in the array
  - Otherwise need a loop

```
>>> myl
[0, 3, 6, -1, 2, 3, 2]
>>> 3 in myl
True
>>> 7 in myl
False
>>>
```

For Loops

For Loop



- Convenient

- Instead of:

```
myl = [...]
cntr = 0
while cntr < len(myl):
    print("Item", myl[cntr])
    cntr = cntr + 1
```

SEQUENCES

Types of Sequence

- Lists and string are similar
  - String never change ('immutable')

- Both are **sequences**
- Other sequences
  - Range: range(0, 10)
  - Tuple: ('hello', 101)

SYLLABUS

Syllabus – Arrays

- GCSE (OCR)
  - Use arrays simply (one dimension)
- AS/A2 (AQA)
  - Arrays of arrays
  - Foundation for data structures
  - Array algorithms: searching, sorting

Summary

- Arrays are 'composite' values
  - Multiple values ...
  - ... one variable

- Essential for programming when number of items vary
  - e.g. Shopping list
  - **Almost always**