Learning Objectives

- Be able to follow and write algorithms and programs that use one-dimensional data structures (string, array, record)
- How to use a for loop and range() function to iterate through a list in python.

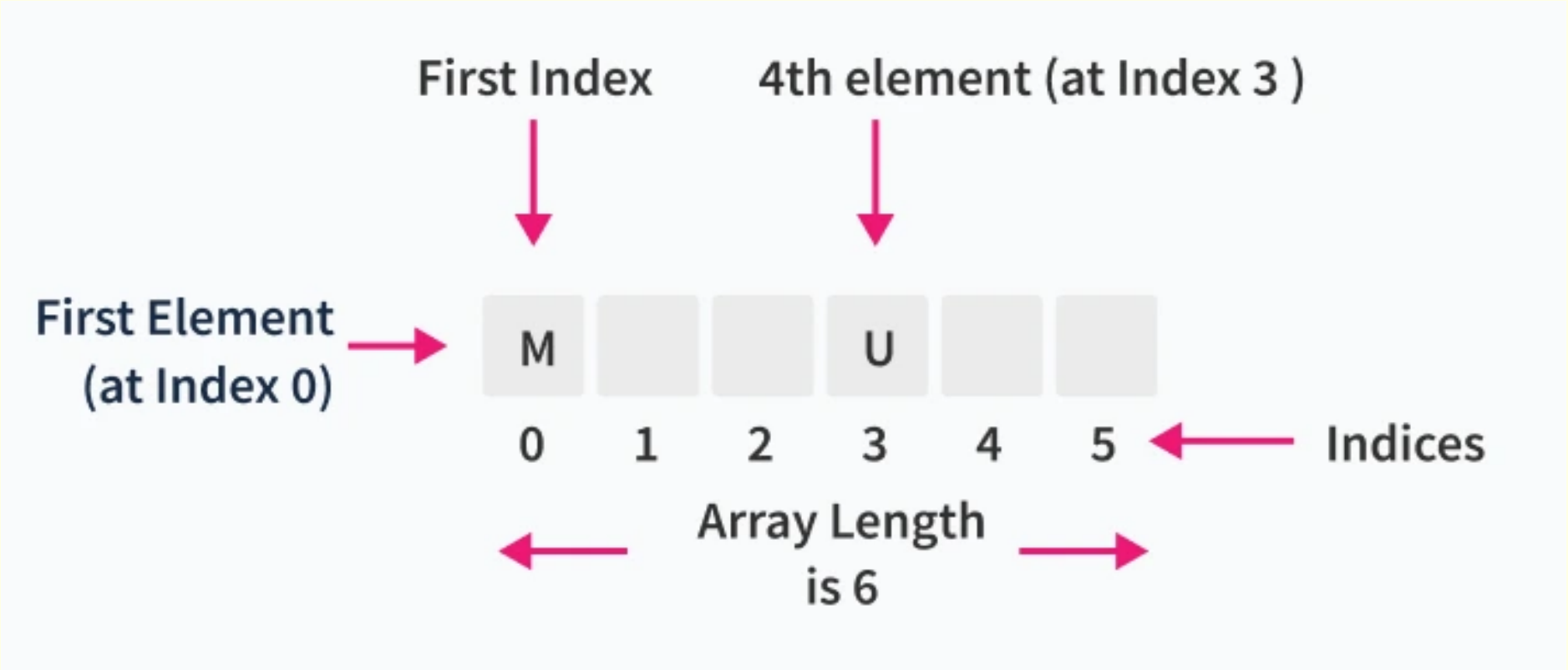
Key Words	
Variable	A location of memory that contains a single item of data under one identifier.
Data Structure	A data organisation and storage construct which can contain multiple data items under one identifier.

List	A dynamic data structure that holds items under one name. The items can be of varying data types.
Array	A fixed (static) data structure that holds items of the same data type under one name.
Index	The location of items or elements in a list, array, or string.
Append	Adding to an existing data structure.
Data structure	Used to store data in an organised and accessible way.
Operator	A symbol or function that performs an operation. For example, +.

What is a data structure?

A data structure is an organised collection of related elements

A 1D data structure is a single list of elements



What is an Array?

- Allows multiple items of data to be stored under one identifier
- **Same data type**
- Can store a table structure
- Reduces need for multiple variables
- Below is an example of a 1D array that stores a collection of colours:

Each element has an index
Starting at 0

0	1	2	3	4	5
blue	**green**	**purple**	**cyan**	**magenta**	**violet**

```
arrayColours = ["blue", "green", "purple", "cyan", "magenta", "violet"]

arrayScores = [75,82,45,55]
```

The benefits of using arrays are:

- Code is easier to follow and therefore easier to debug and maintain
- A group of data items can be easily processed using a for loop
- When you process data held in an array, you typically do the same thing to each data item, so having them stored in numbered locations makes this much easier and quicker to code.

What is a record?

- The record data structure stores a related elements of **different data types.**

- Each element in a record is known as a field.

```
studentRecord = [384, "Collins", "Ivy",2010,15.34]
```

Creating a list and Adding items

An empty list (a list that contains nothing, but can be added to) is created using either one of the
following lines of code:

```
listName = list()
listName = []
```

It is possible to add an item to a list with the code using append. Append will add the item to the end of the list:

```
listName.append("Item")
```

Creates a list named score. It is initialised with 12 zeros.

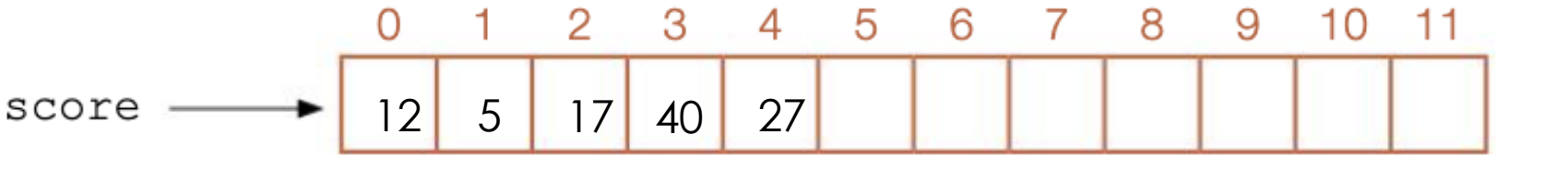
```
score = [0]*12
```

An alternative way to initialise a list would be:

```
score = [0,0,0,0,0,0,0,0,0,0,0,0]
score[0] = 33
```

Example - scores

This program processes 12 numbers using a simple array of integers called score. Imagine a table with one row of 12 boxes:



- The individual boxes in the array can be used just like variables:
- Assign values to them:

```
score[4] = 27
```

- Input values into them from the keyboard or a file:

```
score[4] = int(input("Enter score: "))
```

- Output the value stored in a box to the screen or a file:

```
print("The fourth value is ", score[3])
```

Indexing

Individual items in lists can be accesses using indexing

```
arrayFriends = ["Alice", "Barry", "Catherine"]
arrayScores = [75, 82, 45, 55]
studentRecord = [384, "Collins", "Ivy", 2010, 15.34]
```

Program code	Console output
`print (arrayFriends[1])`	Barry

`print (arrayScores[2])`
`arrayScores[2] = 100`
`print (arrayScores[2])`
	45
	
	100

`print (studentRecord[0] + arrayScores[3])`	439

Finding Length

- Using the built-in function len()

```
arrayFriends = ["Alice", "Barry", "Catherine"]
arrayScores = [75, 82, 45, 55]
studentRecord = [384, "Collins", "Ivy", 2010, 15.34]
```

Program code	Result/output
`len (arrayFriends)`	3
`len (arrayFriends[2])`	9
`len (studentRecord)`	5
`len (studentRecord[1])`	7
`len (emptyList)`	0
`print (arrayScores[len(arrayScores)-1])`	55

Using a for loop to iterate through a list and print the items

```
mySubjects = ["French",
              "Maths",
              "English",
              "Drama",
              "Art"]

for index in range(len(mySubjects))
    print(mySubjects[index])

for subject in mySubjects:
    print(subject)
```

Calculating a total

The following algorithm initialises each element of an array to zero, then gets 12 numbers from the user, adds them up and outputs the total:


```
total = 0
score = [0]*12

for game in range(0,12):
    score[game] = int(input("Enter number: "))
    total = total + score[game]
print("Total is " + str(total))
```

Practice Task

Write a program which asks the user for ten names and inputs them into a list, converts each name to uppercase and prints out the list at the end:

```
names = [""]*10
for i in range(0, len(names)):
    name = input("Enter name: ")
    names[i] = name.upper()
print(names)
```

Alternatively:

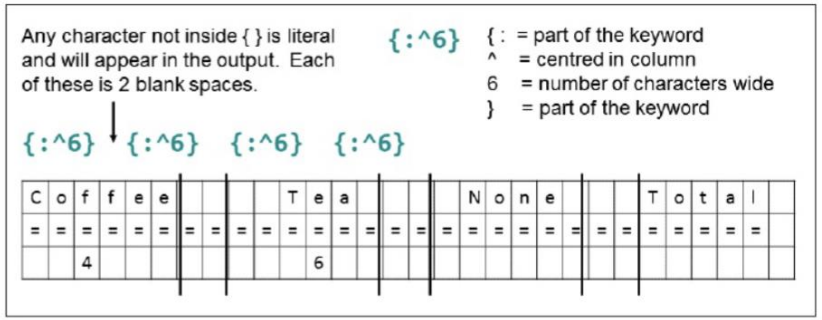
```
names = []
for i in range(0, 10):
    name = input("Enter name: ")
    names.append(name.upper())
print(names)
```

Searching for an item in a list

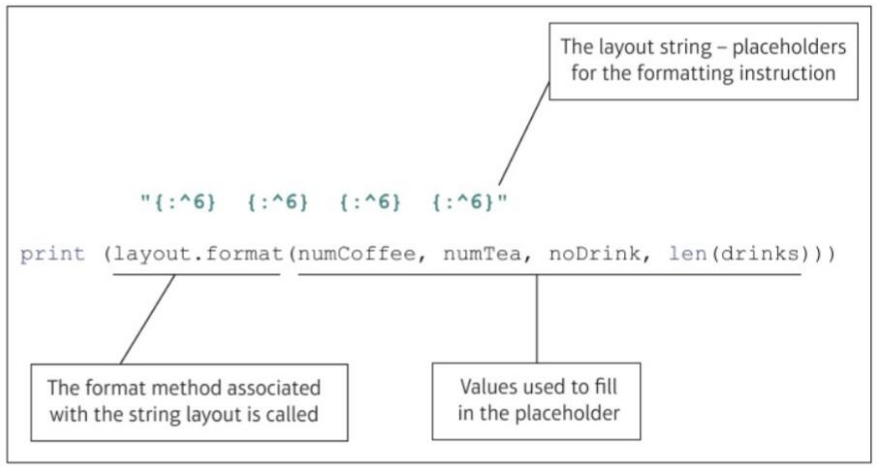
```
index = 0
found = False
Item = input()

while index < len(shopping_list) and not found:
      if item == shopping_list[i] :
      print("Item Found")
               found = True
   else:
      index = index + 1
If not found:
   print("Item not in Shopping list")
```

Formatting Output

```
Coffee      Tea       None      Total
==============================
  4         6         0         10
```



Example	Meaning
`layout = "{:<10}"`	Left justified, 10 wide
`layout = "{:<10d}"`	Left justified, 10 wide, integer
`layout = "{:^10.2f}"`	Centre justified, 10 wide, including 2 decimals, float
`layout = "{:>9.3f}"`	Right justified, 9 wide, including 3 decimals, float



Worked example

This is the same Coffee or Tea program as used above. However, this time the output has been formatted into columns for easier reading.

```
1  # -----------------------------------------------------------------------------
2  # Global variables
3  # -----------------------------------------------------------------------------
4  numCoffee = 0              # For counting up
5  numTea = 0
6  noDrink = 0
7  choice = ""                # What the person wants to drink
8  layout = ""                # For formatting string
9  drinks = ["T", "T", "C", "T", "C", "T", "T","C", "T", "C"]
10
11 # -----------------------------------------------------------------------------
12 # Main program
13 # -----------------------------------------------------------------------------
14
15 for choice in drinks:
16     if (choice == "T"):
17         numTea = numTea + 1
18     elif (choice == "C"):
19         numCoffee = numCoffee + 1
20     else:
21         noDrink = noDrink + 1
22
23 # Layout and print the headers for the columns
24 layout = "{:^6}  {:^6}  {:^6}  {:^6}"
25 print (layout.format("Coffee", "Tea", "None", "Total"))
26
27 # Print a line to divide the headers from the numbers
28 print ("="*30)
29
30 # Print out the line required
31 print (layout.format(numCoffee, numTea, noDrink, len(drinks)))
```

Console session:

```
Coffee      Tea       None      Total
==============================
  4         6         0         10
```