Objective 9:
Understand arrays and lists

In this objective you learn how to store multiple items without having lots of different variables.

Tasks

1. Try entering the following program and see what happens:

```
#Declare list and data set
sentence = ["The","quick","grey","fox","jumps"]
#Output contents of list
print(sentence[0])
print(sentence[1])
print(sentence[2])
print(sentence[3])
print(sentence[4])
```

Sentence is an example of a list, or one dimension array: a group of elements with the same name and data type.

List name:	sentence				
Index:	0	1	2	3	4
Data:	The	quick	grey	fox	jumps

2. Try adding this code, enter the colour brown and see what happens:

```
sentence[2] = input("Enter new colour: ")
print(sentence)
```

Note how it is possible to change an individual element of an array by referring to the index.

3. You don't always want to populate the array with data when it is declared. For example, to declare an array of 100 elements, you would use the alternative syntax:

```
item = []
for counter in range(100):
      item.append(0)
```

You can then put data into the array using the syntax:

```
item[0] = "Alpha"
item[99] = "Omega"
```

Note that indexes always start at zero. So, one hundred elements is 0-99.

4. The power of lists often comes by combining them with iterations.
   Enter this code as an alternative to the first program and notice how the amount of code is much reduced when you use a for loop to output each of the indexes of the list.

```
      sentence = ["The","quick","brown","fox","jumps"]
      for index in range(0,4):
TAB ⟶     print(sentence[index])
```

5. Try entering this program and test by entering 8743 when prompted for a product code:

```
#Declare array and set data
product = ["1262", "Cornflakes", "£1.40", "8743", "Weetabix",→ "£1.20",
"9512", "Rice Krispies", "£1.32"]
product_code=0  #Product code to find
found=False     #Whether product was found in array

#Ask use for the product code to find
product_code=input("Enter the product to find: ")

#Use a loop to cycle through the indexes
counter = 0
found = False
while found == False and counter != 9:
    #Output the product if found
    if product[counter] == product_code:
        print(product[counter + 1])
        print(product[counter + 2])
        found = True
    counter = counter + 1

#Output message if product is not found
if counter == 9:
    print("Product not found")
```

Note how an iteration and Boolean variable can be used to go through all the indexes of an array to find an item.

This is not the best way to write this program, but does illustrate how a number of programming concepts can be combined.

You can do a lot more with lists in Python when you combine them with methods.

6. Try entering the following program and see what happens:

```
#Create a new list
inventory = []
#Add items to the list
inventory.append("torch")
inventory.append("gold coin")
inventory.append("key")

#Remove items from the list
inventory.remove("gold coin")

#Sort the items into alphabetical order
inventory.sort()

#Output all the items in the list
for counter in range (len(inventory)):
    print(inventory[counter])
```

7. Change the program so that a ‘pick axe’ is added to the inventory.

8. Change the program so the inventory is output and the user can choose which item to drop.

Objective 9: Key learning points

Understand arrays and lists

- Lists are **dynamic data structures** because the size of the list changes when the program is running.
- A list has an **identifier**: the name of the array.
- A list has an **index** which points to one element of the list.
- Indexes start at 0.
- A list has multiple **elements**: the data stored in it.
- Lists have **methods** that are operations that can be performed on the list with no additional code written by the programmer.

Objective 9: Key words

=[]

Example code: `x = []`

Creates a single dimension list called x.

Data can be populated into a list using square brackets and commas to separate items:

```
x = [4, 7, 3, 1, 9]
```

Empty lists must be populated before items can be added at specific indexes. For example, to initialise a list of 100 indexes:

```
item = []
for counter in range(100):
      item.append(0)
```

list.append

Example code: `listname.append(item)`

Adds item to the list.

list.insert

Example code: `listname.insert(index,item)`

Inserts item at position index.

list.remove

Example code: `listname.Remove(item)`

Removes item from the list.

list.pop(x)

Example code: `listname.pop(index)`

Removes an item from the list, stored at the index. Indexes start at 0 with the first item.

list.sort

Example code: `listname.sort()`

Sorts all the items in a list.

list.extend

Example code: `listname.extend(y)`

Appends list y to a list.

list.reverse

Example code: `listname.reverse()`

Reverses the order of items in a list.

del

Example code: `del listname[:]`

Removes all the items from a list.

in

Example code: `if x in listname:`

Returns whether the list contains item x.