Programming Sheet 6
‘Arrays’ in Python

**Aims**

<table>
<thead>
<tr>
<th>Section</th>
<th>Aim</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arrays – A variable with many values</td>
</tr>
<tr>
<td>2</td>
<td>While Loops and Arrays</td>
</tr>
<tr>
<td>3</td>
<td>For loops and Arrays</td>
</tr>
<tr>
<td>4</td>
<td>Lists: Other operations</td>
</tr>
<tr>
<td>5</td>
<td>Lists and string: sequence</td>
</tr>
</tbody>
</table>

**Related topics**

- *Topic 6.1 Lists (Arrays) and For Loops*
- *Topic 6.2 Testing*

**Python and Arrays**

A fundamental idea in programming is to collect individual values together into more complex structure. In most languages, **arrays** (and records or classes) are used to do this. An array corresponds to a sequence of locations in a computer’s memory so, while it is fundamental, it is also quite inflexible. Python takes a different approach:

- lists collect values in order
- dictionaries collect values by name or number

Both lists and dictionaries are more flexible (but different) alternatives to arrays. In this sheet we look at **lists**. We look first at the ‘array like’ behaviour of lists and then the behaviour that goes beyond arrays.

**1 Arrays – A Variable with Many Values**

The key ideas of lists / arrays are:

- Length: a value that has multiple separate values
- Indexing: you can extract a value from the array
- Assigning: you can update a single value in the list/array

*Note: the examples in this section are written using the interactive Python shell, which has the prompt `>>>`. You can write them as programs (i.e. in a file which you save) if you prefer, but you may need to add some extra ‘print’ statements.*

**1.1 Accessing Entries in an Array**

**Exercise 1.1** Try the following example where 'shopping' is an array of strings:

```python
>>> shopping = ['meat', 'veg']
```
The example illustrates:

- Getting the length of a list
- Indexing into a list
- Updating an element of a list

**Exercise 1.2** Try a similar example using numbers.

- Create a list of numbers
- Extract some individual numbers from the list
- Replace one entry in the list with a

1.2 **Assigning to an Entry in an Array**

Just as it is possible to change a variable by assigning a new value, so we can assign a value to an item in an array.

**Exercise 1.3** Consider the following:

```
>>> mynum
[1, 2, 3, 4, 5, 6]
```

What is the new value of `mynum` after the assignment? Try this out and be sure you can explain it.

2 **While Loops and Arrays**

Because an array contains multiple values, it is common to process the values in a list. For example, the following program:

```python
nums = [1,2,3,4,5,6]

counter = 0
while counter < len(nums) :
    print("Entry",counter,"has value",nums[counter])
    counter = counter + 1
```

produces the output:

```
Entry 0 has value 1
Entry 1 has value 2
Entry 2 has value 3
Entry 3 has value 4
Entry 4 has value 5
Entry 5 has value 6
```

**Exercise 2.1** Try this program and explain its behaviour.
Exercise 2.2 Here are 2 more examples. Try each out and explain what it does.

| nums = [1,2,3,4,5,6]                      | nums = [1,2,3,4,5,6] |
| sum = 0                                  | sum = 0              |
| counter = 0                              | counter = 0          |
| while counter < len(nums) :              | while counter < len(nums) : |
|     sum = sum + nums[counter]            |     sum = sum + nums[counter] |
|     counter = counter + 1                |     nums[counter] = sum |
| print("Sum=",sum)                       | counter = counter + 1 |
|                                         | print("Nums=",nums) |

Write the following similar programs

- Ask the user for a number. Add this number to each item in a list of numbers.
- Look at each number in a list in turn. Find the largest number.

3 For Loops and Arrays

A 'for' loop is an easy way to apply an operation to each item in an array.

Exercise 3.1 Try the following example:

```python
shopping = ['fish', 'bread']
for s in shopping:
    print(s.upper())
```

The output is:

FISH
BREAD

In this example, 's' is the name of a variable. Each item in the list is assigned to the variable 's' in turn. Any name can be used for the variable; the following program is exactly the same:

```python
shopping = ['fish', 'bread']
for my_shopping_item in shopping:
    print(my_shopping_item.upper())
```

For loops allow you to write some programs more briefly. However, it is always possible to write programs using the standard while loop.

Exercise 3.2 Rewrite the example above using a while loop. Discuss with someone else the pro and cons of the two versions.

4 List: Other Operations

There are lots of functions that do things to lists. Lists in Python are much more flexible than standard arrays (in other languages):

- A list can change length
- Items can be added or removed
- Two lists can be joined.

4.1 Joining List

Two lists can be joined (or 'concatenated') using a '+'.

Exercise 4.1 Try the following examples:
shopping = ['fish', 'bread']
gifts = ['socks', 'CDs']
shopping = shopping + gifts
print(shopping)
The output is: ['fish', 'bread', 'socks', 'CDs']

4.2 Testing Membership
It is useful to be able test if a value is in a list. This is done using 'in'

```python
if 'socks' in shopping:
    print("Yes, on list")
```

Exercise 4.2 Try adding a similar example to a program you have written earlier.

4.3 Append and remove
It is possible to both add and remove items from a list.

- the ‘append’ function to add an item to the end of the list
- the ‘remove’ function to take items out of the list (an error occurs if the item is not on the list).

```python
shopping = ['fish', 'bread']
shopping.append('newspaper')
print(shopping)
The output is: ['fish', 'bread', 'newspaper']. Extending the example:
shopping = ['fish', 'bread']
shopping.append('newspaper')
shopping.remove('bread')
print(shopping)
shopping.remove('bread')
The output is: ['fish', 'newspaper'] but then an error:
ValueError: list.remove(x): x not in list
```

5 Lists and Strings: Sequences

Note: this section is not essential.

You may have noticed that strings and lists are somewhat similar:

- You can index into them
- You can concatenate them

However, string and lists are not the same. Instead they are both example of the more general idea of a sequence. Sequences include:

1. Strings, e.g. “hello”
2. Lists, e.g. [1, 2, 3]
3. Ranges, e.g. range(1, 10)
4. Tuples, e.g. ('loaves', 5)

Note: we only mention ranges and tuples briefly here.

5.1 The ‘list’ Function
This converts other types of sequences to a list. Here are some examples of list applied to a string, a range and a tuple:
5.2 Operations on Sequences

The following table, from the Python documentation, shows some operations on sequences. You can apply them to both lists and strings (and other sequences).

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x in s</td>
<td>True if an item of s is equal to x, else False</td>
</tr>
<tr>
<td>x not in s</td>
<td>False if an item of s is equal to x, else True</td>
</tr>
<tr>
<td>s + t</td>
<td>the concatenation of s and t</td>
</tr>
<tr>
<td>s * n, n * s</td>
<td>n shallow copies of s concatenated</td>
</tr>
<tr>
<td>s[i]</td>
<td>i'th item of s, origin 0</td>
</tr>
<tr>
<td>s[i:j]</td>
<td>slice of s from i to j</td>
</tr>
<tr>
<td>s[i:j:k]</td>
<td>slice of s from i to j with step k</td>
</tr>
<tr>
<td>len(s)</td>
<td>length of s</td>
</tr>
<tr>
<td>min(s)</td>
<td>smallest item of s</td>
</tr>
<tr>
<td>max(s)</td>
<td>largest item of s</td>
</tr>
<tr>
<td>s.index(i)</td>
<td>index of the first occurrence of i in s</td>
</tr>
<tr>
<td>s.count(i)</td>
<td>total number of occurrences of i in s</td>
</tr>
</tbody>
</table>

5.3 Operations on Strings

There are lots of functions that are useful on string. Here are some useful ones, taken from the Python documentation:

**str.find(sub[, start[, end]])**

Return the lowest index in the string where substring `sub` is found, such that `sub` is contained in the slice `s[start:end]`. Optional arguments `start` and `end` are interpreted as in slice notation. Return -1 if `sub` is not found.

**str.islower()**

Return true if all cased characters in the string are lowercase and there is at least one cased character, false otherwise.

**str.isupper()**

Return true if all cased characters in the string are uppercase and there is at least one cased character, false otherwise.

**str.lower()**

Return a copy of the string converted to lowercase.

**str.split([sep])**

Return a list of the words in the string, using `sep` as the delimiter string. If `sep` is given, consecutive delimiters are not grouped together and are deemed to delimit empty strings (for example, `'1,,2'.split(',') returns ['1', ',', '2']`). The `sep` argument may consist of multiple characters (for example, `'1<>2<>3'.split('<>') returns ['1', '2', '3'])). If `sep` is not specified or is None, a different splitting algorithm is applied: runs of consecutive whitespace are regarded as a single separator, and the result will contain no empty strings at the start or end if the string has leading or trailing whitespace.

**str.upper()**

Return a copy of the string converted to uppercase.
Here are some examples.

```python
>>> Greeting="hello world"

>>> print(Greeting.upper())
HELLO WORLD

>>> Greeting
'hello world'

>>> Greeting.split()
['hello', 'world']
```

Notice that using the 'upper' function did not change the value of 'Greeting'. This is an important but subtle point. String in Python are 'immutable': once created the values do not change. Instead, a new string was created and printed.

**Exercise 5.1** Try out more of the string functions.

### 6 Summary

Python lists are introduced in stages.

#### 6.1 Python lists as arrays

The first stage is to focus on the 'array-like' behaviour of lists. The key points are:

- An array is a collection of values.
- One value out of the array can be looked at (term: 'indexed').
- One value can be updated (term: 'assigned to').
- A while loop can be used to go through the array item by item.
- A for loop is a more concise way to write this.

#### 6.2 Python lists – more flexible than arrays

In Python, a list can be used more flexibly. The key ideas are:

- The length of a list can be increased by appending an item to the end.
- An item can be removed from the list.
- Lists and strings are just two examples of sequences. Many functions work for all types of sequence.