Practical Sheet 4
While Statements: Loops

Aims

<table>
<thead>
<tr>
<th>Section</th>
<th>Aim</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Going in Circles</td>
</tr>
<tr>
<td>2</td>
<td>Going in Different Circles</td>
</tr>
<tr>
<td>3</td>
<td>Loops with a condition</td>
</tr>
</tbody>
</table>

Related topics

- **Topic 4.1 While Statements**
- **Topic 4.2 Faults and Debugging**

1 **Going in Circles – Repeating a Statement**

A statement can be repeated using a loop. In this section we look at loops with counters:

```
while condition :
    statement that is executed for as long as condition is true
```

Example: print “*****” lots of times

```
x = 0
while x < 5 :
    print("*****")
x = x + 1
```

A loop with a counter has the following parts

- Counter variable – x in the example
- Initialisation: x = 0
- Loop condition: x < 5
- Change loop counter: x = x + 1
- Body of loop: print("*****")

Exercise 1.1

Here are some examples. Try them out to get the idea of loops.

<table>
<thead>
<tr>
<th>Program</th>
<th>What is Printed?</th>
</tr>
</thead>
</table>
| x = 0
while x <= 5 :
    print("Hello")
x = x + 1                                             |                  |
| counter = 5
while counter > 0 :
    print("12345")
counter = counter - 1                                  |                  |
Program | What is Printed?
--- | ---
```
counter = 1
while 8 < counter :
    print("David")
    counter = counter + 1
```

Exercise 1.2
Complete the following programs to give the pattern shown:

<table>
<thead>
<tr>
<th>Program Outline</th>
<th>Output Expected</th>
</tr>
</thead>
</table>
| ```
cntr = 0
while ... :
    print(...) 
cntr = cntr + 1
``` | Hello world! Hello world! Hello world! Hello world! |
| ```
counter = 6 
while ... : 
    print(...) 
counter = counter ...
``` | Loops repeat Loops repeat Loops repeat Loops repeat |
| ```
... 
while ... :
    ...
    ... = ... + 1
``` | On and on On and on On and on On and on |

2 Going in Different Circles

Changing the Statement that is Repeated
In the examples above, the statement in the loop does not change. This is a bit limiting.
We now look at examples where the statement changes.
Try the following and check that you understand what they do.

Example: print numbers 0 to 9
```
x = 0 
while x < 10 :
    print(x) 
x = x + 1
```
Example: print the seven times table

```python
x = 1
while x <= 7:
    print(x, "* 7 =", x*7)
    x = x + 1
```

Exercise 2.1

The table below shows two programming tasks. An attempted solution is given, but it is wrong in both cases. Look at the solution carefully:

- Try the program out
- Describe the problem – what does it do wrong?
- Change it to create a correct solution.

<table>
<thead>
<tr>
<th>Output Required (user input)</th>
<th>Incorrect Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>What's your name? William</td>
<td>name = input(&quot;What's your name? &quot;)</td>
</tr>
<tr>
<td>Hello W</td>
<td>x = 1</td>
</tr>
</tbody>
</table>
| Hello Wi                     | while x < len(name):
| Hello Will                   |     print("Hello", name[:x])
| Hello Willi                  |     x = x + 1 |
| Hello Willia                 | |
| Hello William                | |

<table>
<thead>
<tr>
<th>Output Required (user input)</th>
<th>Incorrect Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>What's your name? David</td>
<td>name = input(&quot;What's your name? &quot;)</td>
</tr>
<tr>
<td>D</td>
<td>x = 1</td>
</tr>
</tbody>
</table>
| a                            | while x <= len(name):
| i                            |     print(name[x])
| d                            |     x = x + 1 |
| i                            | |

Exercise 2.2

Write a program to behave as follows (user input underlined):

```
What's your name? William
W
i
  l
  l
  i
    a
      m
```

**Hint**: remember that a string can be replicated using *. So `print(" " * 10)` prints a string of 10 spaces.
3 Loop with a Conditions

So far we have looked at loops with a counter. In this section, we consider more general loops.

- In a loop with a counter, the number of 'iterations' (the number of times the program goes round the loop) is determined by the counter value before the loop starts.
- For many problems we want the number of iterations to depend on what happens after the loop starts.

Here is an example program:

```python
print("Enter your name a letter at a time")
print("End with a full stop")
name = ''
complete = False

#Loop inputting each letter until the name is complete
while not complete :
    letter = input("Next letter: ")
    if letter == ".":        
        complete = True      
    else :                  
        name = name + letter
print("Your name is", name)
```

and the result of running the program:

```
Enter your name a letter at a time
End with a full stop
Next letter: B
Next letter: i
Next letter: l
Next letter: l
Next letter: .
Your name is Bill
```

Do you think this program could be implemented using a counter? Discuss this with someone else.

**Exercise 3.1**

Earlier, we meet the evil dictator who hated the letter ‘e’ (and ‘E’ even more) – but he is not good at spelling. Each subject is required to read his/her name letter by letter: if an ‘e’ or ‘E’ occurs they are immediately sent to the dungeons. If there are no hated letters, they are welcomed (by name) to the dictator’s palace. Write a program for the dictator’s use. (Hint: adapt the one above.)
Summary

We introduced loops in three stages:

1. A loop with a counter and no use of the counter variables
2. A loop with a counter variable that is used or tested in the loop to vary the statements executed each time round the loop
3. A loop with a condition that does not use a counter variable.

4.1 Counter Loops

• A very simple use of loops is a loop with a variable that counts.
• The variable is initialised, tested and changed.
• It is possible to count up (e.g. from 0 or 1) or to count down.
• If you forget to change the counter, the loop goes on for ever.
• The number of iteration is already fixed when the loop starts.

4.2 General Loops

• A more general form of loop allows the number of iterations of the loop to be determined after the loop has started.
• D=For example, a boolean variable (i.e. with values True or False) can be used to determine when the loop should stop.