Computing

Lesson 4: The famous `for`

Python programming with sequences of data

Rebecca Franks

Worked Example 1 Iterating over items

This program uses `for` to iterate over a list of dice rolls and print the value of each item in the list.

```
1  rolls = [1, 4, 3, 6]
2  for dice in rolls:
3      print(dice)
```

Worked Example 2 Counting selected items

This program uses `for` to iterate over a list of dice rolls and **count** the number of items with a value greater than 3.

```
1  rolls = [1, 4, 3, 6]
2  count = 0
3  for dice in rolls:
4      if dice > 3:
5          count = count + 1
6  print(count)
```

Worked Example 3 Collecting selected items into a list

This program uses `for` to iterate over a list of dice rolls and **collect** the items with a value greater than 3 into a new list named `selection`.

```
1  rolls = [1, 4, 3, 6]
2  selection = []
3  for dice in rolls:
4      if dice > 3:
5          selection.append(dice)
6  print(selection)
```

Task 1

Step 1

Open this program **oaknat.uk/comp-py-words-1** in Repl.it

```
1 from ncce.data import dictionary
2 nb_words = len(dictionary)
3 print(nb_words, "english words in the list")
```

Line 1 imports the `dictionary`, i.e. the list of words that the program will use. **This is not a standard Python component.** The list has been created specifically to allow you to perform these tasks.

Task 1

Step 2

Extend the program so that it first prompts the user to enter a word length (number of characters), and then iterates over the `dictionary`, i.e. the list of words, and **counts** the number of words of this length.

Tip: Refer to Worked example 2 about counting the number of selected items in a list.

Tip: Use the `len` function to retrieve the length of each word in the dictionary.

`len(string)` e.g. `len("deoxyribonucleic")` e.g. `len(name)`	Returns the length (number of characters) of a string.

Task 1

Here is some example input and output to show how the program **should** run:

Example

Note: Use this example to check your program. This is the output your program should produce when searching for 12-letter words.

The program displays a prompt and waits for keyboard input.	`Length of words to search for:`
The user types a reply.	12
The program displays the number of words of the given length.	`There are 29126 words with 12 letters`

Task 2

Open this program **oaknat.uk/comp-py-words-2** in Repl.it

```
1 from ncce.data import dictionary
```

Extend the program so that it first prompts the user for a string (a piece of text) to search for, and then iterates over the list of words in the `dictionary` and collects the ones that contain this piece of text into a new list.

In the end, the program should display the collected words, one word per line.

Tip: Refer to **Worked example 3** about collecting selected items into a new list. **Worked example 1** should help with displaying the contents of the new list.

Tip: Use the `in` operator to check if a word contains a piece of text.

Task 2

Here is some example input and output to show you how the program **should** work:

Example

The program displays a prompt and waits for keyboard input.

```
Text to search for:
```

The user types a reply.

```
python
```

The program displays the words that contain the particular substring.

```
python          pythonissa
pythoness       pythonist
pythonic        pythonize
pythonical      pythonoid
pythonid        pythonomorph
pythonidae      pythonomorpha
pythoniform     pythonomorphic
pythoninae      pythonomorphous
pythonine       pythons
pythonism
```

Heartbeat

Task

In this activity, you'll make a program that processes real ECG (electrocardiogram) data from a medical database. Your program will go over the data and detect heartbeats.

Step 1

Open this program **oaknat.uk/comp-py-ecg-1** in Repl.it

```
1 from ncce.mitdb_data import load, plot
2 heartbeat_data = load(100)
3 plot(heartbeat_data, 'heartbeats.png')
```

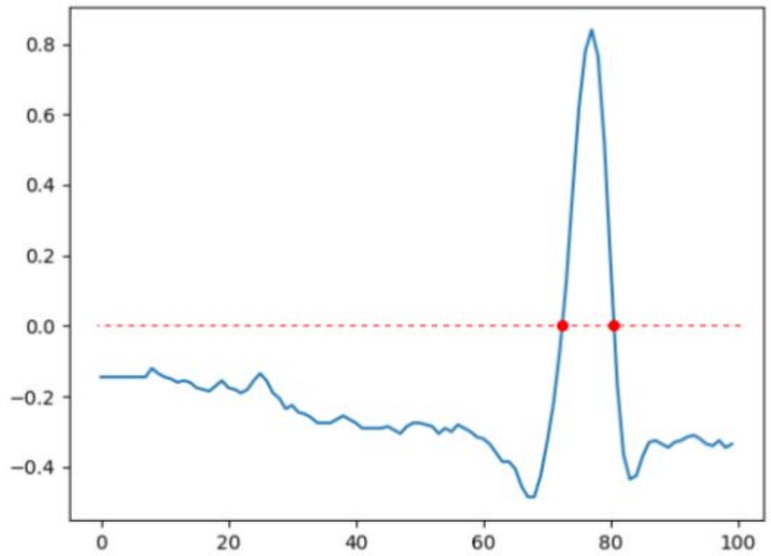
Line 1 imports the `load` and `plot` functions from the `mitdb_data` module. **This is not a standard Python component. It has been created specifically to allow you to perform these tasks.**

The first 100 values will be loaded from the dataset (this is the highlighted 100 in line 2).

Task

Step 2

Run the program. It will create a plot of the loaded values in `heartbeats.png`. You can view this by clicking on the files list on the left hand side.



As you can see, the values are numerical and can range from -1.0 to 1.0. What you see in this plot of the first 100 values is a single heartbeat: values steadily rising over zero, reaching a peak, and then smoothly dropping below zero again.

Task

Step 3

One way to detect a heartbeat is to look for zero crossings in the data (marked with red dots on the image on the previous slide). A zero crossing is a point where values change from positive to negative (or vice versa).

Add the following incomplete code to your program, which will iterate over every value in the data, making sure (in the last line) that the **previous value is also available**.

```
+ previous = -1.0
+ for value in heartbeat_data:
+     
+     Check for
+     crossing .
+
+     previous = value
```

Task

Step 4

Complete the missing instructions in your program (the labeled box), so that your program prints the message "`heartbeat detected`" every time it runs across a value that is positive and its previous value is negative.

Example

Note: Use this example to check your program. This is the output your program should produce for the first 100 values from the data set.

The program displays the message once, i.e. there is one heartbeat in the data loaded.	`heartbeat detected`

Task

Step 5

Modify line 2 in your existing program, so that 1000 data values are loaded from the dataset, instead of 100.

```
from ncce.mitdb_data import load, plot
heartbeat_data = load(1000)
plot(heartbeat_data, 'heartbeats.png')
```


Task

Step 6

Run your program. How many heartbeats is it detecting?

If you want to know if the number of heartbeats detected is correct, check the updated plot of the loaded values in `plot.png`. You can count how many heartbeats are contained in the data.