Lecture 14 – lists, use for in loops to iterate through the elements of a list

At the end of this lecture, students should be able to:

• create a new list
• obtain the length of a list
• use the + operator to concatenate lists
• use the in operator to check if an element is in the list
• iterate through a list using a for...in loop

Recap on for ... in range(...) loops
From lecture 13
• the Python range() function is used to define a sequence of values
• a for...in range() loop can be used to implement counter-controlled repetition

Why lists?
Let’s say we want to store the bank balance amount for every student in this class.

To calculate the total of the first four bank balances?

```
bank01 = 2000
bank02 = 231
bank03 = 21
bank04 = -3000
...
```

To calculate the total of all the bank balances?

```
total = bank01 + bank02 + bank03 + bank04 + bank05 + bank06 + ... + bank632
```

Very awkward!
The list data structure
A list is an ordered sequence of variables (called elements of the list).
Each element of a list has a position in the list, i.e., an index number. The index number always starts at 0.
Each element of a list can be accessed using its index number.
An analogy:

List Syntax – use square brackets
Square brackets are used with lists.
For example, for the following list (named \texttt{my\_list}),

\begin{align*}
\texttt{my\_list} & \rightarrow \\
0 & 12 \\
1 & 21 \\
2 & 34 \\
\ldots & \\
\end{align*}

the element at position 1 in the list can be referred to as \texttt{my\_list[1]}, the first element (at position 0 in the list) can be referred to as \texttt{my\_list[0]}, and so on.

Visualising a list data structure
A list can be visualised:

The elements of a list can be of any type, e.g.,

In reality, each element of a list is a reference (see the two diagrams below):

Creating a list in Python
Square brackets are used to create a list which contains some elements. Each element is separated from the next element using a comma, e.g.,

An empty list (contains no elements) can be created:

Another way to create an empty list is:
Printing a list, the length of a list

Lists can be printed using the print() function:

```python
my_list = [5, 2, 7, 4, 3, 8, 0, 1, 9, 6, -3]
list2 = ['Try', 'something', 'new']
print(my_list)
print(list2)
```

The length of a list is the number of elements currently in the list. The function len() can be used to obtain the current length of a list, e.g.,

```python
my_list
```

```python
[5, 2, 7, 4, 3, 8, 0, 1, 9, 6, -3]
```

```python
list1
```

```python
[ ]
```

```python
len(list1)
```

```python
len(list2)
```

The length of a list is the number of elements currently in the list. The function len() can be used to obtain the current length of a list, e.g.,

```python
#Continuing from the code above
number_of_elements = len(my_list)
```

```python
print(number_of_elements)
```

```python
11
```

```python
0
```

```python
3
```

Concatenating lists

The + operator can be used to concatenate (join) two lists. Concatenation returns a new list containing the elements of the first list followed by the elements of the second list, e.g.,

```python
list1 = ['When', 'all', 'else']
list2 = ['fails,', 'read']
list1 = list1 + list2 + ['the', 'directions']
```

```python
print("1.", list1)
```

```python
my_list
```

```python
[5, 2, 7]
```

```python
my_list = my_list + [3, 5]
```

```python
my_list = my_list + [6]
```

```python
print("2.", my_list)
```

```python
[5, 2, 7, 3, 5, 6]
```

The Python 'in' operator

The Python 'in' operator can be used to test if an element is currently present in a list. True is returned if the element is in the list, False otherwise e.g.,

```python
def search_feedback(num_to_find, a_list):
    if num_to_find in a_list:
        print('It is there')
    elif num_to_find + 1 in a_list or num_to_find - 1 in a_list:
        print('Close!')
    else:
        print('Not even close!')
```

```python
def main():
    my_list = list(range(1, 5))
    search_feedback(-1, my_list)
    search_feedback(5, my_list)

main()
```

Populating a list using the range() function

The Python range() function defines a sequence of integer values within two boundaries (see previous lecture). The range() function can be used to populate a list, e.g.,

```python
my_list1 = list(range(5))
print("1.", my_list1)
```

```python
[0, 1, 2, 3, 4]
```

```python
my_list2 = list(range(10, 20, 3))
print("2.", my_list2)
```

```python
[10, 13, 16, 19]
```

```python
my_list3 = list(range(10, 4, -2)) + list(range(4, 10, 3))
print("3.", my_list3)
```

```python
[10, 8, 6, 4, 7]
```
Accessing elements of a list
Each element in a list can be accessed using its index value. (Remember: square brackets are used with lists).

Note that accessing an element at an index value which doesn't exist in the list gives an index error:

```
def main():
a_list = ['What', 'I', 'didn't', 'expect', 'changed', 'me']
phrase = a_list[1] + ' ' + a_list[4]
print(phrase)
phrase = a_list[0] + ' ' + a_list[4] + ' ' + a_list[5]
print(phrase)
main()
```

```
I changed
What changed me
```

Lists are mutable objects. The elements of a list can be updated.

```
def main():
    my_list = [15, 12, 17, 10, 13, 18]
    print("1.", my_list)

    my_list[0] = my_list[1] + my_list[2]

    length = len(my_list)
    my_list[length - 2] = my_list[length - 1]
    print("2.", my_list)

    my_list[length - 1] = "Bye"
    print("3.", my_list)
main()
```

```
1. [15, 12, 17, 10, 13, 18]
2. [29, 80, 17, 10, 31, 31]
3. [29, 80, 17, 10, 31, 'Bye']
```

Visiting each element in the list
One way of accessing each element of a list is shown below where each element is printed:

```
def main():
    my_list = ['We', 'are', 'not', 'anticipating', 'any', 'emergencies']
    print(my_list[0])
    print(my_list[1])
    print(my_list[2])
    print(my_list[3])
    print(my_list[4])
    print(my_list[5])
main()
```

```
We
are
not
anticipating
any
emergencies
```

This is not a useful way of visiting each element. What if there were 100000 elements in the list?

```
def main():
    my_list = ['No', 'keyboard', 'detected.', 'Press', 'F1', 'to', 'continue']
    for element in my_list:
        print(element)
main()
```

```
No
keyboard
detected.
Press
F1
to
continue
```

```
def main():
    my_list = ['No', 'keyboard', 'detected.', 'Press', 'F1', 'to', 'continue']
    for item in my_list:
        print(item)
    for word in my_list:
        print(word)
```

```
Both these loops on the left do exactly the same job as the loop above.
```
Using lists - example
The following program visits each element of a list. The loop variable (item in this code) is assigned a reference to each element of the list in turn.

```python
def count_items(a_list, max_allowed):
    count = 0
    for item in a_list:
        if item < max_allowed:
            count = count + 1
    return count
def main():
    my_list = list()
    for count in range(500):
        num = random.randrange(1, 500)
        my_list = my_list + [num]
    print(count_items(my_list, 250), "elements are under 250")
main()  # 238 elements are under 250
```

Complete the function 1
Complete the following function which is passed a list of ints as a parameter and returns a new list in which each element is the squared value of the element in the original list.

```python
import random
def get_list_of_squares(a_list):

def main():
    my_list = list()
    for count in range(10):
        my_list = my_list + [random.randrange(1, 10)]
    print("1.", get_list_of_squares(my_list))
    print("2.", my_list)
main()  # 1. [64, 64, 9, 36, 81, 64, 36, 64, 4, 1]
    # 2. [8, 8, 3, 6, 9, 8, 6, 8, 2, 1]
```

Complete the function 2
Complete the print_xs() function which prints a line of characters: an "X" is printed if the corresponding element of the parameter list is True, otherwise a space is printed (see the output of the example below where the elements in indexes 0, 3 and 5 are True).

```python
def print_xs(a_list):

def main():
    print("0123456789")
    my_list = [True, False, False, True, False, True]
    print_xs(my_list)
main()  # 0123456789 X X X
```

Complete the function 3
Complete the start_with_vowel_count() function which returns the number of words in the list which start with a vowel. Assume each word in the list has at least one letter.

```python
def start_with_vowel_count(a_list):
    vowels = "aeiouAEIOU"

def main():
    my_list = ['Nobody', 'goes', 'to', 'that', 'restaurant',
               'because', 'it', 'is', 'too', 'crowded']
    vowel_starters = start_with_vowel_count(my_list)
    print("Start with a vowel", vowel_starters)
main()  # Start with a vowel: 2
Summary

In a Python program:

- a list object can be created
- square brackets are part of the notation used with list
- the length of a list can be obtained using the len() functions
- the + operator is used to concatenate two lists
- the 'in' operator is used to check if an element is in the list
- we can iterate through the elements of a list using a for...in loop

Examples of Python features used in this lecture

```python
def print_section():
    a_list = ['What', 'I', "didn't", 'expect,', 'changed', 'me']
    phrase = a_list[1], a_list[4]
    print(phrase)
    phrase = a_list[0], a_list[4], a_list[5]
    print(phrase)

def get_list_of_squares(a_list):
    count = 0
    square_list = []
    for item in a_list:
        square_list = square_list + [item * item]
    return square_list

def create_list_of_randoms():
    my_list = list()
    for i in range(500):
        num = random.randrange(1, 500)
        my_list = my_list + [num]
```