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

Recap

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

```python
def get_dice_throws_results(num_dice_throws, num_to_check):
    count = 0
    for i in range(num_dice_throws):
        dice = random.randrange(1, 7)
        if dice == num_to_check:
            count += 1
    return count

def main():
    print(get_dice_throws_results(30000, 6), "sixes thrown (out of 30000 throws)
    4947 sixes thrown (out of 30000 throws)
    1 sixes thrown (out of 6 throws)
    99621 sixes thrown (out of 600000 throws)
    print(get_dice_throws_results(6, 6), "sixes thrown (out of 6 throws")
    print(get_dice_throws_results(6, 6), "sixes thrown (out of 6 throws")
    print(get_dice_throws_results(600000, 6), "sixes thrown (out of 600000 throws")
    main()
```

Why lists?

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

```python
bank01 = 2000;
bank02 = 231;
bank03 = 21;
bank04 = −3000;
...
```

- To calculate the total of the first four bank balances?

```python
total = bank01 + bank02 + bank03 + bank04
```

- To calculate the total of all the bank balances?

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

Very awkward!
The list data structure

- A list is a sequence of variables (called elements of the list).
- Each element of a list has an index number. The index number always starts at 0.
- Each element of a list can be accessed using its index number.
- An analogy:

<table>
<thead>
<tr>
<th>A simple variable</th>
<th>A structure with many variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>single_home</td>
<td>many_homes[0], many_homes[1], many_homes[2], ...</td>
</tr>
</tbody>
</table>

Visualising a list data structure

- A list can be visualised:

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>34</td>
<td></td>
</tr>
</tbody>
</table>
```

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

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&quot;first&quot;</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21.5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>True</td>
<td></td>
</tr>
</tbody>
</table>
```

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

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>34</td>
<td></td>
</tr>
</tbody>
</table>
```

Creating a list in Python

- Square brackets are used to create a list containing some elements. Each element is separated from the next element using a comma, e.g.,

```
my_list = [12, 21, 34]
```

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

```
my_list = []
```

- Another way to create an empty list is:

```
my_list = list()
```

Note that list is a special word in Python. It refers to the list data structure and it should not be used as a variable name.
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]
list1 = []
list2 = ['Try', 'something', 'new']
print(my_list)
print(list1)
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
# Continuing from the code above
number_of_elements = len(my_list)
print(number_of Elements)  # 11
print(len(list1))  # 0
print(len(list2))  # 3
```

Concatenating lists

- The **+ operator** can be used to concatenate (join) two lists. Concatenation adds the elements of the second list to the end of the first list, e.g.,

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

```python
my_list = [5, 2, 7]
my_list = my_list + [3, 5]
my_list = my_list + [6]
print(my_list)  # [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!')

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

Populate a list using the range() function

- The Python `range()` function defines a sequence of integer values within a boundaries (from lecture 13).
- The `range()` function can be used to populate a list, e.g.,

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

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

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

1. [0, 1, 2, 3, 4]
2. [10, 13, 16, 19]
3. [10, 8, 6, 4, 7]
Accessing elements of a list

Each element in a list can be accessed using its index number. (Remember: square brackets are used with lists.)

```python
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()
```

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

```python
a_list = ['What', 'I', 'didn’t', 'expect', 'changed', 'me']
print(a_list[6])
```

Assigning new values to elements of the list

Elements of a list can be assigned new values. (Remember: square brackets are used with lists.)

```python
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()
```

Give the output

One way of accessing each element is shown below where each element is printed:

```python
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()
```

Visiting each element in the list

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

- The **for**...**in** structure can be used to iterate through each element in the list structure (in their index order from 0 to the end of the list).

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

    for element in my_list:
        print(element)

    main()
```

```python
for item in my_list:
    print(item)

for word in my_list:
    print(word)
```

Both these loops do exactly the same job as the loop above.

### List example

- The following program processes each element of a list.

```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 i 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

- 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
def get_list_of_squares(a_list):

def main():
    my_list = list()
    for i in range(10):
        my_list = my_list + [random.randrange(1, 10)]
    print("1.", my_list)
    print("2.", get_list_of_squares(my_list))

main()
```

1. [8, 8, 3, 6, 9, 8, 6, 8, 2, 1]  
2. [64, 64, 9, 36, 81, 64, 36, 64, 4, 1]

### Complete the function

- Complete the following function which is passed a list of numbers as a parameter. The function prints the largest and the smallest value in the list. You can assume that there is at least one element in the parameter list.

```python
def print_min_max(a_list):

def main():
    my_list = list()
    for i in range(0, 20):
        my_list = my_list + [random.randrange(1, 20)]
    print_min_max(my_list)

main()
```

[14, 10, 10, 18, 9, 16, 12, 15, 7, 6, 7, 15, 17, 7, 19, 17, 6, 4, 9, 16]

Largest: 19, Smallest: 4
Complete the function

- 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 position 0, 3 and 5 are True).

```python
def print_xs(a_list):
    pass
```

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

```
0123456789
 X  X
```

Complete the function

- Complete the start_with_vowel_count() function which returns a count of all the 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"
    pass
```

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

```
Start with a vowel: 2
```

Summary

- In a Python program:
  - a list object can be created
  - 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)
```

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

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