Lecture 15 – the split() method, updating the elements of lists, lists are mutable objects
Learning outcomes

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

• use the index number to access individual elements of a list
• make changes to the elements of a list
• copy the values of a list
• use the split() method on a string to obtain a list of string objects
• lists are mutable objects
Recap

From lecture 14

• we can iterate through the elements of a list using a for...in loop
• calculations can be done using the values in the elements of a list

```python
def print_xs(a_list):
    for item in a_list:
        if item == True:
            print("X", end="")
        else:
            print(" ", end="")

def main():
    print("01234567890123456789")
    list1 = [True, False, False, True, True, False, True, False, False, False, True, True, False, False, True]
    print_xs(list1)
main()
```
Lists Recap - accessing list elements

The elements of a list can be accessed from the end of the list by using a negative index value.

```python
my_list = [10, 20, 30, 40, 50]
print(my_list[-4])
my_list[-3] = my_list[-1] + my_list[-2]
print(my_list[-3], my_list[1], my_list[-5])
```

```
20
90 20, 10
```
Mutable Objects, Immutable Objects

```python
def main():
    list1 = [6, 4, 7]
    value1 = list1[0]
    value1 = value1 + 3
    print("1.", list1, value1)

    list2 = ['a', 'b', 'c']
    value2 = list2[0]
    value2 = value2.upper()
    print("2.", list2, value2)

    list3 = ['a', 'b', 'c']
    value3 = list3
    value3[0] = value3[0].upper()
    print("3.", list3, value3)

main()
```

String and int objects are **immutable** (look at outputs 1. and 2.).

List objects are **mutable** (look at output 3.).

1. [6, 4, 7] 9
2. ['a', 'b', 'c'] A
3. ['A', 'b', 'c'] ['A', 'b', 'c']
Why does the following not work as intended?

In the following for...in loop, each element of the list is accessed but ...

```
def main():
    a_list = [10, 8, 6, 4, 7]
    print("1.", a_list)
    for number in a_list:
        number = number * 2
        print(number, end=" ")
    print()
    print("3.", a_list)
main()
```

1. [10, 8, 6, 4, 7]
20 16 12 8 14
3. [10, 8, 6, 4, 7]

Note that in the above example, the values of the elements in the list have not changed in any way.

What if the intention was to update the element values in the list?
Updating the elements in the list

The elements in a list can be updated if we assign to each element of the list using the index of the element, e.g.,

```python
def main():
    a_list = [10, 8, 6, 4, 7]
    print("1.", a_list)
    number_of_elements = len(a_list)

    for index in range(number_of_elements):
        a_list[index] = a_list[index] * 2

    print("2.", a_list)

main()
```

1. [10, 8, 6, 4, 7]
2. [20, 16, 12, 8, 14]

Changing a value at an index location updates the element of the list.
def main():
    my_list = [10, 8, 6, 4, 7]
    for index in range(len(my_list)):
        print(index, my_list[index] * 2)

main()
Complete the main() function

Complete the code in the `main()` function which adds 1 to each list element which has odd value.

```python
import random
def main():
    a_list = []
    for index in range(10):
        a_list = a_list + [random.randrange(1, 100)]
    print("1.", a_list)
    print("2.", a_list)
main()
```

1. [69, 98, 7, 92, 13, 9, 27, 36, 96, 46]
2. [70, 98, 8, 92, 14, 10, 28, 36, 96, 46]
Complete the code in the `main()` function which changes the elements starting from index 1 so that each element is the accumulative total of the previous elements (i.e., element 1 is the sum of the element 0 and element 1, element 2 is the sum of element 1 and element 2, etc.).

```python
import random

def main():
    a_list = []
    for num in range(10):
        a_list = a_list + [random.randrange(1, 10)]
    print("1.", a_list)

#write code here

print("2.", a_list)
main()
```

1. [8, 1, 9, 5, 6, 3, 6, 4, 5, 6]
2. [8, 9, 18, 23, 29, 32, 38, 42, 47, 53]
The **split()** method separates a **single string** into a **list of the parts of the string** (the tokens) using the separator defined (inside the parentheses). Each element of the list is a string object. This method can be applied to any string object.

If no separator is defined (as in the code below), whitespace is the default separator, e.g.,

```python
def main():
    phrase = 'The best cure for insomnia is to get a lot of sleep'
    words_list = phrase.split()
    print(words_list[0], words_list[4], words_list[7])

main()
```

The insomnia get
def main():
    prompt = "Enter a line of numbers: 
    line_of_nums = input(prompt)
    list_of_nums = line_of_nums.split()
    for index in range(len(list_of_nums)):
        list_of_nums[index] = int(list_of_nums[index])
    total = 0
    for number in list_of_nums:
        total = total + number
    print("Total:", total)

main()

Enter a line of numbers: 4 6 12 13 9
Total: 44

Enter a line of numbers: 5 -3 6 8 1
Total: 17

Note that split() function breaks a string up into a list of strings.
def split_message(message):
    words = message.split()
    num = int(words[1])
    num = num + 4
    words[2] = num
    a_word = words[0]
    words[0] = a_word[:3]
    print(words[1], words[0], words[2], sep = " ")

def main():
    phrase = 'tuna 4 lunch'
    split_message(phrase)

main()
Assigning a list object to a variable

Python lists are **objects**. When an object is assigned to a variable, the reference (the address) is copied and stored in the variable.

```python
list1 = [1, 2, 3]
list2 = list1
list3 = [1, 2, 3]

print(list1)
print(list2)
print(list3)


print()
print(list1)
print(list2)
print(list3)
```

```
[1, 2, 3]
[1, 2, 3]
[1, 2, 3]
[1, 2, 5]
[1, 2, 5]
[1, 2, 4]
```

```
0 1 2
- - -
1 2 3 5
```

```
0 1 2
- - -
1 2 3 4
```
Do the following two sections of code give the same output? If not, what is the difference in output?

**Code A**

```python
list1 = [1, 2, 3]
list2 = list1

for index in range(len(list1)):
    list2[index] = list1[index] * 2

print("1.", list1)
print("2.", list2)
```

**Code B**

```python
list1 = [1, 2, 3]
list2 = [1, 2, 3]

for index in range(len(list1)):
    list2[index] = list1[index] * 2

print("1.", list1)
print("2.", list2)
```
In a Python program:

• a for … in loop can be used to access each individual element of a list

• a for … in range() loop can be used to make changes to individual element of a list

• a list is an object. Assigning a list to a variable makes a copy of the reference (not a copy of the list).

• lists are mutable objects

• we use the split() method to break a string into a list of strings. The default separator for the split() method is whitespace.
Examples of Python features used in this lecture

```python
def change_list(a_list):
    number_of_elements = len(a_list)
    for i in range(number_of_elements):
        a_list[i] = a_list[i] * 2

def use_lists(list1, list2):
    list3 = []
    for index in range(len(list1)):
        list3 = list3 + [list1[index] + list2[index]]
    return list3

def split_message(message):
    words = message.split()
    print(words[2], words[0])
```