Lists and Mutability

A “list” is an ordered sequence of values

It is a mutable data type organized by an index starting at zero

In computer memory, a list is a “complex type” and is stored indirectly

Review:

- Variables are stored in a separate place from “the heap”
- When we create a variable and assign it a primitive value, it is stored as the variable
- When we create a variable and assign it a complex data type, the address of that data is stored as the variable
- In the heap, the address is assigned its primitive values at indexes
- “Address” and “index” are not interchangeable, and distinguishing between them is key to tracing code that includes mutable data
- Illustration of this process in the slides, or can be watched in Visualize Python

A list is stored as an address, and that address has values at indexes

- Indexes can be positive (counting from the left starting at 0) or negative (counting from the right starting at 1)

Because the values within the list can be modified, it is “mutable”

- We refer to a value within a list as **address[index]**

list_address.append(value) adds an item to the end of the list

list_adress[value_index] = new_value changes a specific value within a list

Lists take the form **[‘value_1’, ‘value_2’, ‘value_3’]** and are indexed:

0 (or -3)	1 (or -2)	2 (-1)
value_1	value_2	value_3

Lists can be concatenated:

```
animals1 = []
print (animals1)
animals1 = ['cow', 'horse']
animals2 = ['dog']
animals3 = animals1 + animals2
print (animals3)
```

Lists can be modified using:

append(value) adds a value to the end of a list

insert(index, value) adds a value at a specific index

del list[index] deletes the value at a specific index

remove(value) deletes a value wherever it appears in the list

list.sort() organizes the list alphabetically a-z or numerically in ascending order

list.reverse() organizes the list alphabetically z-a or numerically in descending order

Examples for Visualize Python:

```
animals = ['cow', 'dog', 'horse']
animals.append('deer')
```

```
print (animals[2])
animals[2] = 'duck'
print (animals[0])
print (animals[-1])
print ('The ' + animals[0] + ' and the ' + animals[2] +  ' sleep in the barn.')
animals.insert(2, 'pig')
print (animals)
animals = ['cow', 'dog', 'horse', 'sheep', 'pig' ]
print(animals)
del animals[3]
print(animals)
animals.remove('horse')
print(animals)
```

Keyword **len(list)** tells us how long a list is
Keyword **in** tells us whether a value is in a list, returns True or False
Slicing from yesterday can be used on both strings and lists

If we have a list inside another list, we can use two indexes to retrieve a value from the internal list:
big_list[big_list_index][internal_list_index]

Exercise in slides:

```
# Create a list of 5 integers, display the maximum number
def get_max(lst):
    result = ""
    lst.sort()
    result = lst[-1]
    return result
int = [1, 5, 2, 7, 3]
print(get_max(int)) # outputs 7
# same problem, more complicated solution
def get_max(lst):
    size = len(lst)
    i = 0
    max_value = 0
    while i < size:
        if lst[i] > max_value:
            max_value = lst[i]
        i += 1
    return max_value
```

***Need to know how to do both, because a test question might say “don’t use sort”

Example 1 from schedule page is to run and be sure you understand, not questions
Example 2:

```
# Write a function to gets a one to seven-digit integer,
# and adds together all of its digits.
```

```
# Then take the sum and do the following computation:
#     (((((sum + 9) * 2) - 4) / 2) - sum)
# note: the computation must be in this exact order
# For example, 1234567, the sum would be 28
#     (((((28 + 9) * 2) - 4) / 2) - sum)
# the result would be 7
#
# you may assume the incoming number does not contain lead zeros

def math_trick(number):
    my_list = list(str(number))
    sum = 0
    i = 0
    size = len(my_list)
    while i < size:
        sum += int(my_list[i])
        i += 1
    return (((((sum + 9) * 2) - 4) / 2) - sum)

print(math_trick(1234567))
print(math_trick(2468975))
print(math_trick(345))
print(math_trick(3))
print(math_trick(93))
```

More examples to be added in future at link
http://www.cs.virginia.edu/~up3f/cs1110/examples/datatype/