Overview: Lists

- List = **ordered** sequence of values
- **Mutable** data type
- Because of the ordering, an element in a list can be referred by its **index**.
- Indices start at **zero**
How are Lists Represented in Memory?

- Primitive types are stored directly
- Complex types (such as lists) are stored indirectly

http://www.pythontutor.com/visualize.html

- Trace through code

```
num = 5
grades = [97, 86, 91, num, 88]
num = 33
big = [23, grades, num, 7]
print(big)
grades[1] = 87
grades.append(6)
big[2] = grades
print(big)
```

- What happens when we assign a variable to a list? (in memory)
  - Only the memory address is assigned; the list is not copied
Tracing through Code with Lists

- **Rule 1**
  - Variables and items on the heap are stored in separate locations.

- **Rule 2**
  - A primitive type is stored directly with its variable.
  - A complex type has its variable store a **memory address**.
    - A memory address refers to a location on the heap where the actual data is stored.

- **Rule 3**
  - Every assignment begins by either creating a variable space (and heap location, if necessary), or emptying out the existing contents of a variable space (**but not the heap**!).
  - Copying either a value or memory address from one box into the other.
  - A variable or memory location must only store either numbers/booleans, or a memory address, **never** the name of a variable.
Tracing through Code with Lists

num = 5
grades = [97, 86, 91, num, 88]
num = 33
big = [23, grades, num, 7]
print(big)
grades[1] = 87
grades.append(6)
big[2] = grades
print(big)
print(grades)

Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Heap</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>num</td>
<td>A100</td>
<td>[23, [97, 86, 91, 5, 88], 33, 7]</td>
</tr>
<tr>
<td>grades</td>
<td>A100</td>
<td>[23, [97, 87, 91, 5, 88, 6], [97, 87, 91, 5, 88, 6], 7]</td>
</tr>
<tr>
<td>big</td>
<td>A200</td>
<td>[97, 87, 91, 5, 88, 6]</td>
</tr>
</tbody>
</table>
Creating Lists

```python
animals = ['cow', 'dog', 'horse']  # create a new list
print(animals)

animals1 = []  # create an empty list
print(animals)

animals1 = ['cow', 'horse']
animals2 = ['dog']
animals3 = animals1 + animals2  # concatenate lists
print(animals3)
```
Accessing Items in Lists

animals = ['cow', 'dog', 'horse']  # create a new list

print (animals[2])  # access a particular item
animals[2] = 'duck'  # update a particular item
print (animals[0])  # indices start from zero
print (animals[-1])  # negative numbers start from the end of the list

print ('The ' + animals[0] + ' and the ' + animals[2] + ' sleep in the barn.')
Adding Items to Lists

```python
animals = ['cow', 'dog', 'horse']  
# create a new list
animals.append('deer')  
# add item to a list
print (animals[2])  
# access a particular item
animals[2] = 'duck'  
# update a particular item
print (animals[0])  
# indices start from zero
print (animals[-1])  
# negative numbers start from the end of the list
print ('The ' + animals[0] + ' and the ' + animals[2] + ' sleep in the barn.')
animals.insert(2, 'pig')
print (animals)
```

- `append()` adds an element to the end of a list
- `insert()` adds an element to a particular position of a list
Removing Items from Lists

animals = ['cow', 'dog', 'horse', 'sheep', 'pig']
print(animals)

del animals[3] # by index
print(animals)

animals.remove('horse') # by item / element
print(animals)

del deletes an element at a particular position
remove() removes a particular element
Length of Lists

animals = ['dog', 'cat', 'bird']
counter = 0
while counter < len(animals):
    print (animals[counter])
    counter = counter + 1
print (animals)

`len()` is a function to return the length of a list (i.e., the number of items in a list)
small = [1, 2, 3]
string = 'CS111x Intro to programming'

print (small.index(2))
print (string.index('n'))
print (string.index('111x'))
print (string.index('1'))

**index()** is a function to return an index of an element

Note: You’ll need to check if the element is in the list before trying to get its index
in

string = 'CS111x Intro to programming'

print ('n' in string)
print ('Intro' in string)
print ('cat' in string)

list = [5, 7, 9, 11, 15]
print (7 in list)
print (3 in list)
print (3 not in list)

in is a keyword and can be used to check if the element is in the list or string before trying to get its index
Slicing and Returning Part of a List with [:]

string = 'CS111x Intro to programming'

print (string)
print (string[7:15])

list = [5, 7, 9, 11, 15]
print (list)
print (list[1:4])
print (list[1:])
print (list[:4])
print (list[:−1])

print (type(list[2:4]))
Sorting and Reversing

animals = ['cow', 'dog', 'horse', 'sheep', 'pig']
animals.sort()
print('sorted animals =', animals)

# another way to print (notice a space after "=")
print('sorted animals = ' + str(animals))

animals.reverse()
print('reversed animals = ', animals)

sort() rearranges the items of a list (in ascending order)
reverse() reverses the order of the items in the list
Two Dimensional List (List of Lists)

list1 = [5, 7, 9, 11, 15]
list_of_lists = [['cow', 'horse'], [list], [4, 5, 6]]

print (list_of_lists[0])
# access a particular list
print (list_of_lists[0][1])
# access a particular item

print(len(list_of_lists))
print(len(list_of_lists[1]))