Title: Notes, Week 8
Page 1 of 2

Name: Zach Caldwell
Class: CS 112-01
Date: Feb. 28, 2017

Cues:

List Data Type

List Declaration Syntax

List Indexing

Extracting / Changing
List Values

List Index Errors

List Dimensions
List Slicing

Getting List Length

List Concatenation &
Replication

Notes:

1. Lists:
   a. Are a data structure that contains multiple ordered values. Syntax examples:
   ```
   >>> list = [1, 2, 536]
   >>> list = [‘a’, ‘cat’, ‘stuff42’]
   >>> list = []
   ```
   b. Can have their items addressed via a zero-based index. Example:
   ```
   >>> list = [‘a’, ‘b’, ‘c’]
   >>> list[0]
   a
   >>> list[1]
   b
   ```
   c. Can have their values extracted or changed via indexing. Example (reusing the above list):
   ```
   >>> x = list[0]
   >>> x
   ‘a’
   >>> list[1] = ‘cat’
   >>> list
   [‘a’, ‘cat’, ‘c’]
   ```
   d. Will throw an error if an index greater than the current item range in the list is requested, or if anything other than an integer is given as an index.

   e. Can be multi-dimensional, eg. `list[0][1]`.

   f. Can be *sliced*, which causes a subset of the contained data to be returned as a new list. Example:
   ```
   >>> spam = ['cat', 'bat', 'rat', 'elephant']
   >>> eggs = spam[2:4]
   >>> eggs
   ['rat', 'elephant']
   ```
   g. Can have their current size (number of values) requested via the `len()` function. Example (using the above spam):
   ```
   >>> len(spam)
   4
   ```
   h. Can be concatenated (combined) and replicated (have their items repeated) using the standard + and `*` operators. Example (using the above eggs and spam):
   ```
   >>> spam + eggs
   ```

Title: Notes, Week 8
Page 2 of 2

Name: Zach Caldwell
Class: CS 112-01
Date: Feb. 28, 2017

Cues: **Notes:**

```
['cat', 'bat', 'rat', 'elephant', 'rat',
'elephant']
>>> eggs * 2
['rat', 'elephant', 'rat', 'elephant']
```

The del keyword

i. Can have items removed using the `del` keyword. Example:

```
>>> del eggs[1]
>>> eggs
['rat']
```

list() casting

j. Can be created from multiple values or converted from a tuple using the `list()` cast.

Summary/Reflection: