Lists Part I

- **<u>Sequence</u>: an object that contains multiple items of data**
  - The items are stored in sequence one after another

- **Python provides different types of sequences, including lists and tuples**
  - The difference between these is that a list is mutable and a tuple is immutable

Working with Lists

- Can index list from **front** (first: [0]) or from **back** (last: [-1])

Indices	0	1	2	3	4	5
	‘Alice’	**‘Bob’**	**‘Kala’**	**‘Kamal’**	**‘Laila’**	**‘Terrence’**
Indices	-6	-5	-4	-3	-2	-1

- len(*list_name*): returns length of list
  - ***# of elements, not index of last character!***
- We access elements of a list through its indices. Indices are very important to learn!

Lists

List contains multiple items, enclosed in ***square brackets***

Examples:

```
even_numbers = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
friends = [“Alice”, “Bob”, “Kala”, “Kamal”, “Laila”, “Terrence”]
data = [47.4, 63.9, 33.2, 45.56, 98.6]
mixedData = [1, “Alicia”, 47.3, “Y”, 0, 16, -3.4]
```

Lists

<u>Element</u>: An item in a list

Format: `list = [item1, item2, etc.]`

`print` function can be used to display an entire list

Things you can do with lists 1

- Repeat operator: list * n
  - Makes new list that repeats the old list **n** times.
  - E.g. [1, 2, 3] * 4   ➔   [1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]

- Concatenate operator: list1 + list2
  - Makes new list that consists of list1 followed by list2
  - E.g., [1, 2, 3] + [4, 5, 6, 7]   ➔   [1, 2, 3, 4, 5, 6, 7]

- Get length of list: len(list) (i.e. # of elements in the list
  - E.g., len([‘a’, ‘b’, ‘c’, ‘d’, ‘e’])   ➔   5

Things you can do with lists 2

- Use list index to access elements, e.g.:

```
numbers = [25, 50, 48, 64, 130, 49]
print(numbers[0], numbers[3], numbers[4])
```

- Note: List index starts at zero (0), not 1!
- Assign new values to list members, e.g.:

```
aList = [2, 3, 4, 5, 6, 7]
aList[3] = 42
```

- Use negative numbers to index from end
  - aList[-1] ➔ 7 aList[-3] ➔ 5
  - aList[-len(aList)]

Things you can do with lists 3

- Use with **while** loop

```
names = ["Sue", "Ann", "Sally", "Jill", "Kamala"]
i = 0
while i < len(names):
    print(names[i])
    i = i + 1
```

- Use with **for** loop

```
numbers = [5, 6, 7, 8, 9]
for n in numbers:
   print(n*2)
```

Lists and for loops - By item and by index

Please compare and contast the different kinds of lists with for loops.

See How To Think Like A Computer Scientist’s

[Lists and for loops](http://interactivepython.org/courselib/static/thinkcspy/Lists/Listsandforloops.html)

http://interactivepython.org/courselib/static/thinkcspy/Lists/Listsandforloops.html

Lists and for loop By item

```
fruits = ["apple", "orange", "banana", "cherry"]

for afruit in fruits:     # by item

    print(afruit)
```

```
apple

orange

banana

cherry
```

Lists and for loop By index

```
fruits = ["apple", "orange", "banana", "cherry"]

for position in range(len(fruits)):     # by index

    print(fruits[position])
```

```
apple

orange

banana

cherry
```

Things you can do with lists 4

- Test if item is in list, e.g.:

  aList = [2, 3, 4, 5, 6, 7]

  5 ***in*** aList ➔ **True**

- Test if item is NOT in list, e.g.:

  aList = [2, 3, 4, 5, 6, 7]

  42 ***not in*** aList ➔ **True**

Things you can do with lists 5

- Create sublists using ***slice***, e.g.:

```
oddNumbers = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
smallList = oddNumbers[3 : 6]
bigList = oddnumbers[2:]
```

List slicing format: *`list[start : end+1]`*

(remember start indexing from 0)

From Gaddis:
list_name[start : end]
In the general format, start is the index of the first element in the slice, and end is the index marking the end of the slice. The expression returns a list containing a copy of the elements from start up to (but not including) end .

Things you can do with lists 6

- Assign multiple items using slice

```
>>>listA = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
>>>listA[2:6] = [1, 3, 5, 9]
>>>listA
[2, 4, 1, 3, 5, 9, 14, 16, 18, 20]
```

- Remove multiple items by assigning []

```
>>>listA = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
>>>listA[2:6] = []
>>>listA
[2, 4, 14, 16, 18, 20]
```

Things you can do with lists 7

- Remove items with ***del*** statement

```
>>>fooList = [1, 2, 3, 4]
>>>del fooList[1]
>>>print (fooList)
[1, 3, 4]
```

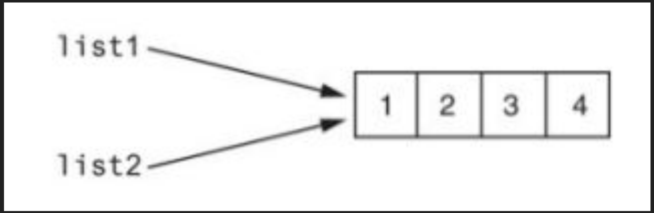
- del statement can use a slice

```
>>>fooList = [1, 2, 3, 4, 5, 6]
>>>del fooList[1:3]
>>>print (fooList)
[1, 4, 5, 6]
```

Important Facts About Lists

Assigning one list to another makes both variables point to *same* list!

```
>>>list1 = [1, 2, 3, 4]
>>>list2 = list1
>>>list2[2] = 42
>>>list1
[1, 2, 42, 4]
>>> list1==list2
True
```



Important Facts About Lists

You could use slice to copy a list (or concatenate) - two separate lists!

```
>>>listA = [1, 2, 3, 4]
>>>listB = listA[:]
>>>listB[2] = 42
>>>listA
[1, 2, 3, 4]
>>>listB
[1, 2, 42, 4]
>>>listB == listA
False
```