Python LIST

MS. RASHMI CHAUDHARY
AP
CSE DEPARTMENT

Lists

- A list is a comma-separated sequence of items, enclosed in square brackets
- Lists can be heterogeneous – items don't have to be from the same data type
- Like strings, lists can be sliced, indexed, concatenated, and repeated.
- The `len()` function will return the number of elements in the list

Lists

- Unlike strings, lists are mutable (can be changed by element assignment)
  - Make an assignment, using the index

```
>>> myList = ['milk','eggs','bread']
>>> myList[1] = 'butter'
>>> myList
['milk', 'butter', 'bread']
```

- You can also assign to slices, and even change the length of the list

List Slice Operations

```
#create a list and assign to a variable
>>> data = ['bob', 32, 'sue', 44]
>>> data
['bob', 32, 'sue', 44]
#assign to a list slice
>>> data[1:3] = ['dave', 14]
>>> data
['bob', 'dave', 14, 44]
#insert an element (or several)
>>> data[1:1] = [19]
>>> data
['bob', 19, 'dave', 14, 44]
#delete an element
>>> data[3:4] = []
>>> data
['bob', 19, 'dave', 44]
```

Python Lists

```
>>> a = [1, 2, 3]
>>> b = a
>>> b
[1, 2, 3]
>>> a[1] = 6
>>> b
[1, 6, 3]
>>> a = [6, 7, 8]
>>> b[2] = 'x'
>>> b
[1, 6, 'x']

>>> # copy or reference semantics?
```

But …

```
>>> a = [6, 7, 8]
>>> b
[1, 6, 'x']
>>> # ???
```

Python Lists

```
>>> x = 13
>>> L1 = [x, 'y', 3]
>>> L1
[13, 'y', 3]
>>> x = 19
>>> L1
[13, 'y', 3]
```

Adding to a List

- You can grow the list dynamically with the concatenation operator:

```
>>> x = [2, 4, 6, 8]
>>> x
[2, 4, 6, 8]
>>> x = x + [10]
>>> x
[2, 3, 6, 8, 10]
```

List Insertion by Slicing

```
>>> x = [2,3,6,8.10]
>>> x
[2, 3, 6, 8.1]
>>> x = x[:2]+[100,200] + x[2:]
>>> x
[2, 3, 100, 200, 6, 8.1]
>>> x = x[:4] + []
>>> x
[2, 3, 100, 200]
```

Nested Lists

```
>>> grades = [100, 97, 85]
>>> stRec = ['A000','jack',grades]
>>> stRec
['A000', 'jack', [100, 97, 85]]
>>> len(stRec)
3
```

Additional String Operations

- `split`: divides a string into a list of substrings

```
>>> myStr = 'The fat black cat'
>>> myStr.split()
['The', 'fat', 'black', 'cat']
```

- `split` defaults to blank as the delimiter, but you can specify a different character:

```
>>> myStr = '12/10/2008'
>>> myStr.split('/')
['12', '10', '2008']
```

Strings

- After you have split a string into a list of substrings, you may want to convert some of the substrings to specific data types.
  - specific casts: *int( ), float( ), long( ),* and str( )
  - If you don't know the data types, you can use the generic cast *eval( )*

Example

```
>>> mysplitStr
['12', '10', '2008']
>>> first = eval(mysplitStr[0])
>>> first
12
>>> #etc. - you can use the type function to
  determine if the list elements are the
  type you expected:
>>> x = 3.4
>>> if type(x) == float:
  print('float')

float
```

More About Lists

- See Section 5.1 in the Python tutorial to get a whole set of list functions:
  - append(x)
  - insert(i, x)
  - etc.
- Since lists are objects, dot notation is used to call the functions
- When using string functions you may need to import the string library:
  - **`import string`**

List Functions

```
>>> stRec = ['A000', 'jack', grades]
>>> stRec.remove(grades)
>>> stRec
['A000', 'jack']
>>> stRec.append([100, 97, 85])
>>> stRec
['A000', 'jack', [100, 97, 85]]
>>> stRec.pop(2)#removes item at index
[100, 97, 85]
>>> stRec
['A000', 'jack']
```