Lists Part II and Tuples

Review lists and for loops

Please compare and contast the different kinds of lists with for loops - this is an important one to learn!

See How To Think Like A Computer Scientist's

Lists and for loops 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
```

List Methods

- <list name>.<method name>(<argument>)
- append(item) #appends item to list
- index(item) #returns index of item
- count(item) #returns count of item
- insert(index, item) # inserts item at index
- sort() #sorts list (ascending order)
- remove(item) #removes item from list
- reverse() #reverses order of list
- pop(index) #removes & returns item (default: last)

List Methods:
Append vs. Concatenate

- myList.append(newItem) modifies the list
  - No need to assign the result to a variable
  - myList.append(newItem) #correct form
  - x = myList.append(newItem) #assigns ***none*** to x
- result = myList + [newItem] #concatenate
  - Result lost unless you assign it to a variable
    myList + newItem
  - Can assign back to original list ➔ accumulator pattern, e.g.,
    myList = myList +[newItem]

List Accumulators

```
nameList = []
name = input("Enter a name: ")
while name != '':
    nameList.append(name)
    # nameList = nameList + [name]     # alternative
    name = input("Enter a name: ")
print(nameList)
```

List Comprehensions

- [<expression> for x in <list>]
- Examples:

  [item**2 for item in [1,2,3,4,5,6,7,8,9,0]]

  [x % 2 == 0 for x in [1,2,3,4,5,6,7,8,9,0]]

  [c.upper() for c in ["a", "b", "c", "d", "e"]]

- [<expression> for x in <list> if <boolean>]

  [x**2 for x in [1,2,3,4,5,6,8,10] if x%2==0]

List Min and Max

- min(list): minimum (smallest) item
- max(list): maximum (largest) item

Lists and Files

readlines()

readlines() reads the contents of the whole file into a ***list*** (including \n)

```
infile=open('cities.txt', 'r')
cities=infile.readlines()
infile.close()
print(cities)
```

readlines() with rstrip(‘\n’)

```
infile=open('cities.txt', 'r')
cities=infile.readlines()
infile.close()

index=0
while index < len(cities):
    cities[index] = cities[index].rstrip(‘\n’)
    index+=1
print(cities)
```

Writing from List to File Method I writelines()

writelines() writes an entire list to a file

but no spaces between words!

```
cities=[‘New York’, ‘Boston’, ‘San Francisco’, ‘Los Angeles’]
outfile=open(‘cities.txt’, ‘w’)
outfile.writelines(cities)
outfile.close()
```

Writing from List to File Method II for loop with write()

```
cities=[‘New York’, ‘Boston’, ‘San Francisco’, ‘Los Angeles’]
outfile=open(‘cities.txt’, ‘w’)

for item in cities:
    outfile.write(item + ‘\n’)
outfile.close()
```

Using join() to join elements into a String

```
cities=[‘New York’, ‘Boston’, ‘San Francisco’, ‘Los Angeles’]

#join a list with an empty space
print(' '.join(cities))
```

New York Boston San Francisco Los Angeles

```
#join a list with a comma
cities_string=(', '.join(cities))

print(cities_string)
```

New York, Boston, San Francisco, Los Angeles

```
type(cities_string)
```

Writing from List to File with integers

```
numbers=[1,2,3,4,5,6]
outfile=open('numberlist.txt', 'w')
for item in numbers:
    outfile.write(str(item) + '\n')
outfile.close()
```

Reading in integers from a file to a list

```
infile=open('numberlist.txt', 'r')
numbers=infile.readlines()
infile.close()  #try this first

index=0
while index<len(numbers):
    numbers[index] = int(numbers[index])
    index+=1

print(numbers)
```

Tuples: Like Lists but *Immutable*

- List:

  ```
  x = [1, 2, 3, 4, 5]
  x[2] = 9            #OK
  ```

- Tuple () not []:

  ```
  z = (1, 2, 3, 4, 5)
  z[2] = 9            #Not OK: Error!
  ```

Tuples support same operations as lists except those that change the contents of the list.

Why do tuples exist?

- ➔ processing a tuple is faster than processing a list.
- ➔ tuples are safe -> cannot be modified.
- ➔ certain operations in Python that require the use of a tuple

  (Tuples are also comparable and hashable e.g. we can use tuples as key values in Python dictionaries but not lists.)

Converting between Lists and Tuples

list()

tuple()