Data Structures in Python

- Although Python does not require explicit type declarations, it provides us with several useful data structures:
  - Lists
  - Tuples
  - Sequences
  - Sets
  - Dictionaries
Lists

• We have already seen that lists in Python allow us to do most of the things for which we use arrays in other languages.

• List operations in Python include:
  - append
  - insert
  - pop
  - count
  - reverse
  - extend
  - remove
  - index
  - sort

append

• **append** adds an item to the end of the list

• Syntax:

  ```python
  list.append(x) - adds x to the end of list.
  ```

• Example:

  ```python
  a.append(x)
  ```

  is equivalent to

  ```python
  a[len(a):] = [x]
  ```
extend

- **extend** extends the list by appending all the items in another list.
- Syntax:
  ```python
  list.extend(anotherList) – adds every member of anotherList to list
  ```
- Example
  ```python
  a.extend(b)
  ```
  is equivalent to
  ```python
  a[len(a):] = b
  ```

insert

- **insert** places an item at a specific position in the list.
- Syntax:
  ```python
  list.insert(index, x) – adds x to list before element index of the list
  ```
- Example:
  ```python
  a.insert(0, x) is placed at the beginning of a
  a.insert(len(a), x) is placed at the end of a
  ```
**remove**

- **remove** removes the first item with a specified value from the list.

- Syntax:
  
  `list.remove(x)` – removes the first occurrence of `x` from `list`.

- Example:
  
  ```
a = ['Hello', 'Goodbye', 'Au revoir', 'Hello']
a.remove('Hello') – removes the first "Hello"
```
**pop() – An Example**

```
SIEGFRIE@panther:~/python$ cat stacky.py
a = [-1, 4,  66.25, 333, 333, 1, 1234.5]

x = a.pop(4)
print x

y = a.pop()
print y
print a
SIEGFRIE@panther:~/python$ python stacky.py
333
1234.5
[-1, 4, 66.25, 333, 1]
SIEGFRIE@panther:~/python$
```

**index**

- `index(x)` returns the index in the list of the first item whose value is `x`.
- It is an error if there is no such item.
- You can use the construct
  ```python
  if x in a
  or
  if x not in a
  ```
  to determine if the item `x` is in list `a`. 
index() – an Example

SIEGFRIE@panther:~/python$ cat listy.py

```python
a = [-1, 4, 66.25, 333, 333, 1, 1234.5]
x = a.index(4)
print x

if 41 in a:
    y = a.index(41)
    print y
else:
    print "41 is not in the list"

if 65 not in a:
    print "There\'s no 65."

print a
```

SIEGFRIE@panther:~/python$ python listy.py

```
1
41 is not in the list
There\'s no 65.
[-1, 4, 66.25, 333, 333, 1, 1234.5]
```

SIEGFRIE@panther:~/python$
count()

- **count**(x) returns the number of times that x appears in the list.

**count() – an Example**

```
SIEGFRIE@panther:~/python$ cat howlisty.py
a = [-1, 4, 66.25, 333, 333, 1, 1234.5]

x = a.count(333)
print x

y = a.count(4)
print y
z = a.count(-2)
print z
SIEGFRIE@panther:~/python$ python howlisty.py
2
1
0
SIEGFRIE@panther:~/python$
```
sort

- `sort()` sorts the items of the list in place.
- `sort()` can also take parameters, allowing a user to specify their own methods to determine sorting order, the key field in a structure and reverse the order of the sort.

sort() – An Example

```
SIEGFRIE@panther:~/python$ cat sort.py
a = [66.25, 333, 333, 1, 1234.5]

print a
a.sort()
print a

b = ["Robert", "robert", "c c cummings",\
   "will i am", "William Tell"]

print b
b.sort()
print b
```
SIEGFRIE@panther:~/python$ python sort.py
[66.25, 333, 333, 1, 1234.5]
[1, 66.25, 333, 333, 1234.5]
['Robert', 'robert', 'c c cummings', 'will i am', 'William Tell']
['Robert', 'William Tell', 'c c cummings', 'robert', 'will i am']
SIEGFRIE@panther:~/python$

reverse

- **list.reverse()** reverses the elements of the list, in place.
- Example
  SIEGFRIE@panther:~/python$ cat reverse.py
  a = [66.25, -1, 333, 1, 1234.5, 333]
  print a
  a.reverse()
  print a
  SIEGFRIE@panther:~/python$ python reverse.py
  [66.25, -1, 333, 1, 1234.5, 333]
  [333, 1234.5, 1, 333, -1, 66.25]
  SIEGFRIE@panther:~/python$
Using Lists as Stacks

• The list methods make it very easy to use a list as a stack, where the last element added is the first element retrieved (“last-in, first-out”).
• To add an item to the top of the stack, use append().
• To retrieve an item from the top of the stack, use pop() without an explicit index.

Using Lists as Stacks

SIEGFRIE@panther:~/python$ cat lstack.py
stack = [3, 4, 5]
stack.append(6)
stack.append(7)
print stack

x = stack.pop()
print "x popped off the stack is ", x
print "The stack is now ", stack

y = stack.pop()
print "y popped off the stack is ", y
z = stack.pop()
print "z popped off the stack is ", z
print "The stack is now ", stack
SIEGFRIE@panther:~/python$ python lstack.py
[3, 4, 5, 6, 7]
x popped off the stack is 7
The stack is now [3, 4, 5, 6]
y popped off the stack is 6
z popped off the stack is 5
The stack is now [3, 4]
SIEGFRIE@panther:~/python$

Using Lists as Queues

• It is also possible to use a list as a queue, where the first element added is the first element retrieved ("first-in, first-out"); however, lists are not efficient for this purpose.

• While appends and pops from the end of list are fast, doing inserts or pops from the beginning of a list is slow (because all of the other elements have to be shifted by one).
Using Lists as Queues

• To implement a queue, use collections.deque which was designed to have fast appends and pops from either side.

Using Lists as Queues – An Example

SIEGFRIE@panther:~/python$ cat queue.py
from collections import deque
queue = deque(["Eric", "John", "Michael"])  
queue.append("Terry")  # Terry arrives
queue.append("Graham")  # Graham arrives
name = queue.popleft()  # The first to arrive
                        # now leaves
print "The name is ", name

name = queue.popleft()  # The second to arrive
                        # now leaves
print "The next name is ", name
print queue              # Remaining queue in
                        # order of arrival
deque(['Michael', 'Terry', 'Graham'])
print "The queue is now ", queue
SIEGFRIE@panther:~/python$ python queue.py
The name is Eric
The next name is John
deque(['Michael', 'Terry', 'Graham'])
The queue is now deque(['Michael', 'Terry', 'Graham'])
SIEGFRIE@panther:~/python$