List

- General purpose
- Most widely used data structure
- Grow and shrink size as needed
- Sequence type
- Sortable

Tuple

- Immutable (can't add/change)
- Useful for fixed data
- Faster than Lists
- Sequence type

Set

- Store non-duplicate items
- Very fast access vs Lists
- Math Set ops (union, intersect)
- Unordered

Dict

- Key/Value pairs
- Associative array, like Java HashMap
- Unordered

SEQUENCES (String, List, Tuple)

- indexing: x[6]
- Slicing: x[1:4]

- adding/concatenating:
- multiplying:
- checking membership:
- iterating

in/not in for i in x:

- len(sequence1)
- min(sequence1)
- max(sequence1)
- sum(sequence1[1:3]])
- sorted(list1)

- sequence1.count(item)
- sequence1.index(item)

Indexing: access any item in the sequence using it’s index.

Slicing: slice out substrings, subtuples using indexes [start : end+1 : step]

Adding/substracting: combine 2 sequences of the same type using +

Multiplying: multiply a sequence using *

Checking membership: test whether an item is in or not in a sequence

Iterating: iterate through the items in a sequence

Number of items: count the number of items in a sentence

Minimum: - Find the minimum item in a sequence lexicographically
- alpha or numeric types, but cannot mix types

Maximum: - Find the maximum item in a sequence
- alpha or numeric types, but cannot mix types

Sum: - Find the sum of items in a sequence
- entire sequence must be numeric type

Sorting: - Returns a new list of items in sorted order
- Does not change the original list

Count (item): - Returns count of an item

Index item: - Returns the index of the first occurrence of an item

Unpacking: - Unpack the n items of a sequence into n variables

LISTS
All operations from Sequences, plus:
• constructors:

• del list1[2]	delete item from list1
• list1.append(item)	appends an item to list1
• list1.extend(sequence1)	appends a sequence to list1
• list1.insert(index, item)	inserts item at index pops last item
• list1.pop()	removes first instance of item reverses list order sorts list in place
• list1.remove(item)	
• list1.reverse()	
• list1.sort()	

constructors - creating a new list

x = list ()
x= l'a', 25, ' dog', 8.43]
x = list (tuplel)
List Comprehension:
x = [m for m in range (8)]
resulting list: [0, 1, 2, 3, 4, 5, 6, 71
x = [z**2 for z in range (10) if z>4]
resulting list: [25, 36, 49, 64, 81]

TUPLES

• Support all operations for Sequences
• Immutable, but member objects may be mutable
• If the contents of a list shouldn't change, use a tuple to prevent items from accidently being added, changed or deleted
• Tuples are more efficient than lists due to Python's implementation

constructors - creating a new tuple

```
X = ()                  # no-item tuple
x = (1,2,3)
x = 1, 2, 3             # parenthesis are optional
x = 2,                  #single-item tuple
x = tuple (list1)       # tuple from list
```