Tuples
Summary of Common List Operations

- **myList[i]**: the \( i^{\text{th}} \) element of \( \text{myList} \)
- **myList[i:j]**: a sublist of \( \text{myList} \) consisting of the elements from \( i \) to \( j \) of \( \text{myList} \)

- **list1 + list2**: concatenates two lists \( \text{list1} \) and \( \text{list2} \)
- **myList * n**, \( n * \text{myList} \): \( n \) copies of \( \text{myList} \) concatenated together
- **len(myList)**: returns the number of elements in \( \text{myList} \)
- **min(myList)**: returns the smallest element in \( \text{myList} \)
- **max(myList)**: returns the largest element in \( \text{myList} \)
- **sum(myList)**: returns the sum of all elements in \( \text{myList} \)

- **<, <=, >, >=, ==, !=**: used to compare two lists

- **x in myList**: True if \( x \) is an element of the list \( \text{myList} \)
- **x not in myList**: True if \( x \) is not an element of the list \( \text{myList} \)
Summary of Common Tuple Operations

- \( t[i] \)  
  the \( i \)th element of \( t \)

- \( t[i:j] \)  
  a slice of \( t \) consisting of the elements from \( i \) to \( j \) of \( t \)

- \( t1 + t2 \)  
  concatenates two tuples \( t1 \) and \( t2 \)

- \( t*n, n*t \)  
  \( n \) copies of \( t \) concatenated together

- \( \text{len}(t) \)  
  returns the number of elements in \( t \)

- \( \text{min}(t) \)  
  returns the smallest element in \( t \)

- \( \text{max}(t) \)  
  returns the largest element in \( t \)

- \( \text{sum}(t) \)  
  returns the sum of all elements in \( t \)

- \( <, <=, > >, =, ==, != \)  
  used to compare two tuples

- \( x \text{ in } t \)  
  True if \( x \) is an element of the tuple \( t \)

- \( x \text{ not in } t \)  
  True if \( x \) is not an element of the tuple \( t \)
Dictionaries
Summary of Common Dictionary Operations

`del dict[key]`  delete the key/value pair associated with key
`len(dict)`  returns the number of elements in dict

`x in dict`  True if x is an element of dict
`x not in dict`  True if x is not an element of dict

Methods:

`dict.clear()`  Deletes all entries from dict and returns None.
`dict.get(key)`  Returns the value corresponding to the key
  • **Same as** `dict[key]` except if the key isn’t present, it returns None instead of an error
`dict.pop(key)`  Removes the key/value pair & returns the value
  • **Same as** `del dict[key]` except it returns a value
Summary of Common Dictionary Operations

These three methods return sequences that are useful for looping:

- `dict.keys()`: Returns a sequence of the keys
- `dict.values()`: Returns a sequence of the values
- `dict.items()`: Returns a sequence of tuples containing the keys

```python
for key in d2.keys():  # iterates through the keys
    # code...
for value in d2.values():  # iterates through the values
    # code...
for item in d2.items():  # iterates through key/value tuples
    # code...
```

Using simultaneous assignment to split up the tuples:

```python
for key, value in d2.items():
    # code...
```
Sets
Summary of Common Set Operations

Since sets are not ordered, there is no indexing or slicing of sets.

- `len(mySet)` returns the number of elements in `mySet`
- `min(mySet)` returns the smallest element in `mySet`
- `max(mySet)` returns the largest element in `mySet`
- `sum(mySet)` returns the sum of all elements in `mySet`

- `x in mySet` True if `x` is an element of the list `mySet`
- `x not in mySet` True if `x` is not an element of the list `mySet`

- `mySet.add(item)` add an item to `mySet`
- `mySet.remove(item)` remove an item from `mySet`
Subsets and Supersets

A set \( s_1 \) is a \textit{subset} of \( s_2 \) if every element of \( s_1 \) is also in \( s_2 \).

We also say that \( s_2 \) is a \textit{superset} of \( s_1 \).

Set \( s_1 \) is said to be a \textit{proper subset} of set \( s_2 \) if \( s_1 \) is a subset of \( s_2 \), but \( s_1 \neq s_2 \).

Set \( s_2 \) is said to be a \textit{proper superset} of set \( s_1 \) if \( s_2 \) is a superset of \( s_1 \), but \( s_1 \neq s_2 \).

\[
s_1.\text{issubset}(s_2) \quad \text{returns True if } s_1 \text{ is a subset of } s_2, \text{ False otherwise}
\]

\[
s_2.\text{issuperset}(s_1) \quad \text{returns True if } s_2 \text{ is a superset of } s_1, \text{ False otherwise}
\]

\[
s_1 < s_2 \quad \text{returns True if } s_1 \text{ is a proper subset of } s_2
\]

\[
s_1 \leq s_2 \quad \text{returns True if } s_1 \text{ is a subset of } s_2
\]

\[
s_1 > s_2 \quad \text{returns True if } s_1 \text{ is a proper superset of } s_2
\]

\[
s_1 \geq s_2 \quad \text{returns True if } s_1 \text{ is a superset of } s_2
\]
Set Operations

The **union** of two sets is a set that contains all of the elements from both sets.

\[ s1 \cup s2 \]
\[ s1 \cup s2 \]

The **intersection** of two sets is a set that contains the elements that appear in both sets.

\[ s1 \cap s2 \]
\[ s1 \cap s2 \]

The **difference** between two sets is a set that contains the elements in the first set that are not in the second set.

\[ s1 \setminus s2 \]
\[ s1 - s2 \]

The **symmetric difference** (or exclusive or) between two sets is a set that contains the elements that are in exactly one of the two sets.

\[ s1 \Delta s2 \]
\[ s1 ^ s2 \]