CSCE 110: Programming I

David Kebo Hougninou

Basics of Python

Collective Data Structures
lists, sets, dictionaries, tuples
Sets

A set is a collection of distinct and unordered objects.
Sets are mutable. (We can add or remove items from them)
Elements of sets are immutable.
Elements of sets are unique.
Sets cannot be indexed.

Sets support a standard collection of operations:
  • union
  • intersection
  • difference
  • symmetric difference

The Language of Sets

Let S be the set of all integers between 1 and 5, inclusive:
\[ S = \{1,2,3,4,5\} \]
We want to determine if x is an element of set S: \( x \in S \)?
Let \( x = 3 \); is \( x \in S \)?
\( \text{Yes, } x \in S \)

Now let \( x = 10 \); is \( x \in S \)?
\( \text{No, so } x \notin S \)
Common Number Sets

\[ C = \text{set of complex numbers} \]
\[ R = \text{set of real numbers} \]
\[ R^+ = \text{set of positive real numbers} \]
\[ Q = \text{set of rational numbers} \]
\[ Z = \text{integers} = \{…,-3,-2,-1,0,1,2,3,…\} \]
\[ Z^+ = \text{positive integers} = \{1,2,3,…..\} \]
\[ N = \text{natural numbers} = \{0,1,2,3….\} \]
Creating sets

To create a set, place all the elements inside curly braces {}, separated by comma.

The elements can be heterogeneous: integer, float, tuple, string etc.

A set cannot contain mutable elements like lists, sets or dictionaries.

```
# set of integers
numbers = {1, 2, 3, 4}
print(numbers)

# heterogeneous datatypes
numbers = {3.14, "Howdy", (3, 0, 110)}
print(numbers)
```

Creating sets: set() constructor

A set can be created by using the built-in function set().

We can create a set from another object such as a list.

```
# A set cannot have duplicates
numbers = {1, 2, 3, 4, 3, 2}
print(numbers)

# Make a set from a list
numbers = set([1, 2, 3, 2])
print(numbers)
```
Sets

a is a set, b is a list, c = set(b) creates a set from list b

```
>>> a = {1, 4, 'apple', 2, 3}
>>> a
{1, 2, 3, 4, 'apple'}
>>> b = ['apple', 3, 'banana', 3]
>>> b
['apple', 3, 'banana', 3]
>>> c = set(b)
>>> c
{3, 'apple', 'banana'}
```

Sets: empty set

The command set() creates an empty set.

```
> a = set()
> a
set()
```
Sets: add() and update()

Sets are **mutable** and **unordered**.

The **add()** method adds a **single** element.

The **update()** method adds **multiple** elements (tuples, lists, strings, sets)

```
roster = {1, 3}
print(roster)
roster.add(2)
print(roster)
roster.update([2, 3, 4])
print(roster)
roster.update([4, 5], {1, 6, 8})
print(roster)
```

```
>>> [evaluate sets_add_update.py]
{1, 3}
{1, 2, 3}
{1, 2, 3, 4}
{1, 2, 3, 4, 5, 6, 8}
```

Sets: discard() and remove()

The **discard()** method removes an item from a set.

The **remove()** method removes an item from a set, it raises an error if the item does not exist.

```
roster = {1, 3, 4, 5, 6}
print(roster)
roster.discard(4)
print(roster)
roster.remove(6)
print(roster)
```

```
>>> [evaluate sets_remove_discard.py]
{1, 3, 4, 5, 6}
{1, 3, 5, 6}
{1, 3, 5}
```
Sets: pop() and clear()

The `pop()` method removes and returns an element.
An element can be popped arbitrarily from a set.
The `clear()` method removes all elements from the set.

```python
roster = set("Howdy")
print(roster)
print(roster.pop())
roster.clear()
print(roster)
```

```
$python sets_pop_clear.py
{'y', 'o', 'd', 'w', 'H'}
y
set()
```
**Sets Union**

The **union** of sets A and B is the set of all elements that are in **at least one** of A or B. Denoted as $A \cup B$

![Sets Union Diagram](image1)

**Sets Intersection**

The **intersection** of sets A and B is the set of all elements that **common to both** A or B. Denoted as $A \cap B$

![Sets Intersection Diagram](image2)
Sets Difference

The difference \( B \) minus \( A \) is the set of all elements that are in \( B \) and not \( A \). Denoted as \( B - A \).

Symmetric Difference

The Symmetric Difference of \( A \) and \( B \) is a set of elements in both \( A \) and \( B \) except those that are common in both.
Set Complement

The complement of A is the set of all elements in U that are not in A. Denoted as $A^c$.

Formal Definitions of Set Operators

$$A \cup B = \{x \in U \mid x \in A \lor x \in B\}$$
$$A \cap B = \{x \in U \mid x \in A \land x \in B\}$$
$$B - A = \{x \in U \mid x \in B \land x \notin A\}$$
$$A^c = \{x \in U \mid x \notin A\}$$
Set: Union and Intersection

Union is done using the \( \cup \) operator or the `union()` method. Interception is done using the \( \cap \) operator or the `intersection()` method.

```python
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}
print(A | B)       # Sets Union
print(A.union(B))  # Sets Union
print(A & B)       # Sets Intersection
print(A.intersection(B))  # Sets Intersection
```

Set: Difference and Symmetric Difference

Difference is done using the \( - \) operator or the `difference()` method. Symmetric difference is done using the \( ^\) operator or the `symmetric_difference()` method.

```python
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}
print(A - B)         # Sets Difference
print(A.difference(B))  # Sets Difference
print(A ^ B)          # Symmetric Difference
print(A.symmetric_difference(B))  # Symmetric Difference
```

```python
> [evaluate sets_operations.py]
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
{4, 5}
{4, 5}
{1, 2, 3}
{1, 2, 3}
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
```
Sets: symmetric difference

\[ A \triangle B = (A \cup B) - (A \cap B) \]

Subsets

A set A is a subset of set B if every element of A is in B. \( A \subseteq B \).
\[ a.issubset(c) \]

```
>>> a
{1, 2, 3, 4, 'apple'}
>>> c
{'apple', 'banana', 3}
>>> a.symmetric_difference(c)
{1, 2, 4, 'banana'}
```
Other set methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>difference()</td>
<td>Returns the difference of two or more sets as a new set</td>
</tr>
<tr>
<td>difference_update()</td>
<td>Removes all elements of another set from this set</td>
</tr>
<tr>
<td>discard()</td>
<td>Removes an element from the set if it is a member</td>
</tr>
<tr>
<td>intersection()</td>
<td>Returns the intersection of two sets as a new set</td>
</tr>
<tr>
<td>intersection_update()</td>
<td>Updates the set with the intersection of itself and another</td>
</tr>
<tr>
<td>isdisjoint()</td>
<td>Returns True if two sets have a null intersection</td>
</tr>
<tr>
<td>issubset()</td>
<td>Returns True if another set contains this set</td>
</tr>
<tr>
<td>issuperset()</td>
<td>Returns True if this set contains another set</td>
</tr>
<tr>
<td>symmetric_difference()</td>
<td>Returns the symmetric difference of two sets as a new set</td>
</tr>
<tr>
<td>symmetric_difference_update()</td>
<td>Updates a set with the symmetric difference of itself and another</td>
</tr>
<tr>
<td>union()</td>
<td>Returns the union of sets in a new set</td>
</tr>
</tbody>
</table>

Sets: built-in functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all()</td>
<td>Return True if all elements of the set are true.</td>
</tr>
<tr>
<td>any()</td>
<td>Return True if any element of the set is true.</td>
</tr>
<tr>
<td>enumerate()</td>
<td>Return an enumerate object. It contains the index and value of all the items of set as a pair.</td>
</tr>
<tr>
<td>len()</td>
<td>Return the length of the set.</td>
</tr>
<tr>
<td>max()</td>
<td>Return the largest item in the set.</td>
</tr>
<tr>
<td>min()</td>
<td>Return the smallest item in the set.</td>
</tr>
<tr>
<td>sorted()</td>
<td>Return a new sorted list from elements in the set.</td>
</tr>
<tr>
<td>sum()</td>
<td>Return the sum of all elements in the set.</td>
</tr>
</tbody>
</table>