Sharif University of Technology
Department of Computer Engineering

Fundamentals of Programming

Python Language

Arman Malekzadeh
PhD Candidate in Artificial Intelligence

Table of contents

Sets

Sets in Python

- A set is an unordered collection of items. Every element is unique (no duplicates) and must be immutable (which cannot be changed).
- However, the set itself is mutable. We can add or remove items from it.
- Sets can be used to perform mathematical set operations like union, intersection, symmetric difference etc.

```
my_set = {1, 2, 3}
my_set = {1.0, "Hello", (1, 2, 3)}
```

Sets in Python

- We can make set from a list using **set()** function.
- Sets cannot have mutable elements like lists, sets or dictionaries as its elements.
- We can make set from a string.
- Duplicates are not allowed.

```
# set from list
my_set = set([1, 2, 3, 2])
print(my_set) # prints {1, 2, 3}
# set from string
my_set = set("Hello")
print(my_set) # prints {'H', 'e', 'l', 'o'}
```

Set Comprehensions

- In Python, set comprehensions are similar to list comprehensions. The only difference between them is that set comprehensions use curly brackets **{}**.

```
# set comprehension
my_set = {x for x in 'Hello'}
print(my_set) # prints {'H', 'e', 'l', 'o'}
```

```
my_set = {x for x in 'HelloWorld' if x not in 'low'}
print(my_set) # prints {'H', 'r', 'd', 'W', 'e'}
```

Adding Elements to Sets

- We can add single element using the **add()** method and multiple elements using the **update()** method.
- The **update()** method can take tuples, lists, strings or other sets as its argument. In all cases, duplicates are avoided.

```
my_set = {1, 3} # initialize my_set
print(my_set) # prints {1, 3}
my_set.add(2) # add an element
print(my_set) # prints {1, 2, 3}
my_set.update([2, 3, 4]) # add multiple elements
print(my_set) # prints {1, 2, 3, 4}
my_set.update([4, 5], {1, 6, 8}) # add list and set
print(my_set) # prints {1, 2, 3, 4, 5, 6, 8}
```

Removing Elements from Sets

- We can remove elements from a set by using **discard()** and **remove()** methods.
- The only difference between the two is that, while using **discard()** if the element does not exist in the set, it remains unchanged. But **remove()** will raise an error in such condition.

```
my_set = {1, 3, 4, 5, 6} # initialize my_set
print(my_set) # prints {1, 3, 4, 5, 6}
my_set.discard(4) # discard an element
print(my_set) # prints {1, 3, 5, 6}
my_set.remove(6) # remove an element
print(my_set) # prints {1, 3, 5}
my_set.discard(2) # discard an element
print(my_set) # prints {1, 3, 5}
my_set.remove(2) # remove an element
print(my_set) # prints KeyError: 2
```

Removing Elements from Sets

- We can also use the **pop()** method to remove an item. But this method will remove only the last element. Remember that sets are unordered, so you will not know what item that gets removed.

- The following example will illustrate this.

```
# initialize my_set
my_set = set("HelloWorld")
print(my_set)
# pop an element
print(my_set.pop())
print(my_set)
# pop another element
print(my_set.pop())
print(my_set)
```

Removing Elements from Sets

- We can also use the **clear()** method to empty a set.

- The following example will illustrate this.

```
# initialize my_set
my_set = set("HelloWorld")
print(my_set)
# clear my_set
my_set.clear()
print(my_set)
```

The Union of Sets

- The **union()** method returns a new set with all items from both sets.

- The following example will illustrate this.

```
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}
# union of A and B
print(A.union(B)) # prints {1, 2, 3, 4, 5, 6, 7, 8}
print(B.union(A)) # prints {1, 2, 3, 4, 5, 6, 7, 8}
```

The Intersection of Sets

- The **intersection()** method returns a new set with items that are common to both sets.

- The following example will illustrate this.

```
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}
# intersection of A and B
print(A.intersection(B)) # prints {4, 5}
print(B.intersection(A)) # prints {4, 5}
```

References

References I

[1] B Downey, A. (2015). Think Python: How to Think Like a Computer Scientist-2nd Edition.

[2] Deitel, H. M., & Deitel, P. J. (2004). C: How to program. Pearson Educacion.

Sharif University of Technology
Department of Computer Engineering

Arman Malekzadeh

Fundamentals of Programming

Python Language