

1811/2807/7001ICT

Programming Principles

School of Information and Communication Technology
Griffith University

Trimester 1, 2024

23 Python's Class set

Python's set class is a powerful collection type.

23.1 Class set

A Python `set` is a collection of values, where **the values are unique** (no duplicates).

A `set` is *mutable*, meaning we can add and remove objects to/from a set.

But to be able to ensure that the objects in a `set` are unique, and will always stay unique, they must be *immutable*.

So `sets` may contain, numbers, strings, `bools`, `tuples`, but not `lists` or `sets` (or `dicts`).

23.1.1 Empty sets

The *only* way to write an empty set in Python is to call the constructor.

You might have guessed `{}` would make an empty set, but because dictionaries are used more often, and they also use braces, `{}` makes an empty dict.

```
>>> set()
set()
>>> {}
{}
>>> type({})
<class 'dict'>
>>>
```

23.1.2 Non-empty sets

Non-empty sets may be formed:

- with set literals, using braces;

```
>>> {1, 3, 'Andrew'}  
{1, 3, 'Andrew'}  
>>>
```

- by adding elements to existing sets;

```
>>> a = {1, 2}  
>>> a.add(5)  
>>> a  
{1, 2, 5}  
>>>
```

- using the `set` constructor with another collection or sequence as its argument; and

```
>>> b = set([1, 2, 2, 2, 3])  
>>> b  
{1, 2, 3}  
>>>
```

- with set comprehensions.

```
>>> {i for i in range(10) if i % 3 == 0}  
{0, 9, 3, 6}  
>>>
```

Note the weird order of the elements in the last example. The order of elements in a set is not important or preserved.

23.1.3 Set operations

These are things that can be done with sets.

<i>Python</i>	<i>math</i>	<i>description</i>
<code>set()</code>	$\{\}$	Make an empty set.
<code>set(sequence)</code>		Make a set from the values in the sequence.
<code>s.add(x)</code>		Add x to s .
<code>s.remove(x)</code>		Remove x from s .
<code>len(s)</code>	$\#S$	Count the number of elements in s .
<code>x in s</code>	$x \in S$	Is x an element of s ?
<code>x not in s</code>	$x \notin S$	Is x not an element of s ?

<i>Python</i>	<i>math</i>	<i>description</i>
<code>s.union(t)</code> , or <code>s t</code>	$S \cup T$	Form the union of <code>s</code> and <code>t</code> .
<code>s.intersection(t)</code> , or <code>s & t</code>	$S \cap T$	Form the intersection of <code>s</code> and <code>t</code> .
<code>s.difference(t)</code> , or <code>s - t</code>	$S - T$	Form the difference of <code>s</code> and <code>t</code> .
<code>s.issubset(t)</code> , or <code>s <= t</code>	$S \subseteq T$	Is <code>s</code> a subset of <code>t</code> ?
<code>s < t</code>	$S \subset T$	Is <code>s</code> a proper subset of <code>t</code> ?
<code>s.issuperset(t)</code> , or <code>s >= t</code>	$S \supseteq T$	Is <code>s</code> a superset of <code>t</code> ?
<code>s > t</code>	$S \supset T$	Is <code>s</code> a proper superset of <code>t</code> ?

<i>Python</i>	<i>math</i>	<i>description</i>
<code>s.isdisjoint(t)</code>	$S \cap T = \{\}$	Are <i>s</i> and <i>t</i> disjoint?
<code>s.pop()</code>		Return and remove a random element from <i>s</i> .
<code>s.clear()</code>		Remove all elements from <i>s</i> .

There are also some shorthand assignment operators for sets: `|=`; `&=`; and `-=`.

23.2 Set example

This is a classic problem that needs a `set`.

Problem: Write a program that prompts for and reads a file name, then reads the file, and prints all the distinct words in the file, in ascending order.

Hint: use the `sorted` built-in function.

Section summary

This section covered:

- The `set` class, set literals, and operations on sets.