1 sets

Sets are unordered collections of unique values. Common uses include membership testing, finding overlaps and differences between sets, and removing duplicates from a sequence.

Sets, like dictionaries, are defined using curly brackets.

1.1 Creating sets

Creating a set directly:

```
my_set = {'a','e','i','o','u'}
print (my_set)

OUT:
{'a', 'i', 'u', 'o', 'e'}
```

Creating a set from a list:

```
my_list = ['a','e','i','o','u']
my_set = set(my_list)
print (my_set)

OUT:
{'a', 'i', 'u', 'o', 'e'}
```

Creating an empty set. This cannot be done simply with

```
my_set = set()
print (my_set)
```

1.2 Adding to a set and removing duplicates

Adding one element to a set:

```
my_set.add('y')
print (my_set)

OUT:
{'a', 'y', 'i', 'u', 'o', 'e'}
```

Adding multiple elements to a set (note that the set has no duplication of entries already present; sets automatically remove duplicate items):

```
my_set.add('y')
print (my_set)

OUT:
{'a', 'y', 'i', 'u', 'o', 'e'}
```

1.3 Analysing the intersection between sets

Sets may be used to explore the intersection betwen sets. There are usually two orms of syntax which may be used, both of which are given in the examples below.

```
set1 = {'a','b','c','d','e','f'}
set2 = {'a','e','i','o','u'}
```

The difference between two sets:

```
print ('difference')
print (set1.difference(set2))
print (set1 - set2)

OUT:
difference
{'c', 'f', 'b', 'd'}
{'c', 'f', 'b', 'd'}
```

The intersection between two sets:

```
print ('Intersection')
print (set1.intersection(set2))
print (set1 & set2)

OUT:
Intersection
{'a', 'e'}
{'a', 'e'}
```

Is a subset? Is set 2 wholly included in set 1?

```
print ('Is subset?')
print (set1.issubset(set2))
print (set1 <= set2)

OUT:
Is subset?
False
False
```

Is a superset? Does set 1 wholly include set 2?

```
print ('Is superset?')
print (set1.issuperset(set2))
print (set1 >= set2)

OUT:
Is superset?
False
False
```

The union between two sets:

```
print ('Union')
print (set1.union(set2))
print (set1 | set2)

OUT:
Union
{'b', 'o', 'a', 'f', 'c', 'i', 'u', 'd', 'e'}
{'b', 'o', 'a', 'f', 'c', 'i', 'u', 'd', 'e'}
```

Symmetric difference: in either but not both (equivalent to xor)

```
print ('Symmetric difference')
print (set1.symmetric_difference(set2))
print (set1 ^ set2)

OUT:
Symmetric difference
{'c', 'i', 'b', 'u', 'f', 'o', 'd'}
{'c', 'i', 'b', 'u', 'f', 'o', 'd'}
```