Sets: The basics

Sets are the fundamental building block of modern mathematics. Intuitively, a set is a collection of objects, but with two important twists:

1. Sets are unordered.
2. Sets contain no duplicates.

Example 1 Suppose you want to keep a record of which words occur in a text. You aren't interested in how often a given word occured, just whether it occurs at all. Nor do you care in which order the words occurred in the text. So you are actually interested in the *set* of words that occur in the text.

```
# Converting a text to the set of words
import re

def text_to_set(text):
    return set(re.findall(r"\w+", text.lower()))

# change the string below as you see fit
text = "If police police police, then police police police."
print("The original text is:")
print(text)
print("The set of words is:")
print(text_to_set(text))
```

Each property is explained in detail below, but let's first put some helpful notation in place.

List notation

Sets are often written as lists with curly braces around them. So $\{a, b, c, d\}$ denotes the set containing a, b, c, d. Here a, b, c, d are some arbitrary objects. This is known as **list notation**. More complex sets are defined with **set-builder notation**, which will be covered in a later unit.

Example 2 Consider the string *If John slept, then Mary left*. Its set of words (ignoring sentence-initial capitalization) is $\{\text{if}, \text{John}, \text{left}, \text{Mary}, \text{slept}, \text{then}\}$.

Exercise 1 Write the following as a set:

1. the first names of your three favorite actors/actresses,
2. the colors of the rainbow,
3. all prime numbers between 1 and 10 (remember, 1 is not a prime number!)

Elements and set membership

The objects contained in a set are called its **elements** or **members**. One writes $e \in S$ to indicate that e is an element of S. The opposite is denoted $e \notin S$: e is not an element of S. The symbol $\in$ thus indicates **set membership**.

Example 3 Let W be the set of words in the string *If John slept, then Mary left*. Then it holds that *left* $\in W$ and *right* $\notin W$. But it is not the case that *then* $\notin W$ or *awake* $\in W$.

Sometimes $\ni$ is used as the mirror image of $\in$. For example, $a \in S$ could also be written as $S \ni a$.

Example 4 Continuing the previous example, it is true that *left* $\in W \ni$ *then*. That is to say, both *left* $\in W$ and *then* $\in W$ are true.

Exercise 2 Put $\in$, $\ni$, $\notin$, $\not\ni$ in the gaps below as appropriate:

1. $5\_\{1,2,4,5,8\}$
2. $6\_\{1,2,4,5,8\}$
3. $\{5\}\_\{1,2,4,5,8\}$
4. $5\_\{1,2,4,5,8\}\_6$

Lack of order

Even though we may write sets in a linear fashion as lists, they have no internal order. The set $\{a,b\}$ could also be written as $\{b,a\}$. So we have $\{a,b\} = \{b,a\}$, and $\{a,b,c\} = \{a,c,b\} = \{b,a,c\} = \{b,c,a\} = \{c,a,b\} = \{c,b,a\}$.

Example 5 Consider the strings *If John slept, then Mary left* and *If Mary left, then John slept*. While they are clearly distinct sentences, their sets of words are identical.

```python
import re

def text_to_set(text):
    return set(re.findall(r"\w+", text.lower()))

text1 = "If John slept, then Mary left."
text2 = "If Mary left, then John slept."

set1, set2 = text_to_set(text1), text_to_set(text2)
print("Are the sets identical?")
print("Yes") if set1 == set2 else print("No")
```

Exercise 3 For each one of the following, fill the gap with $=$ or $\neq$ as appropriate:

1. $\{a,b\}\_\{a,b\}$

2. $\{b, a\}$ _ $\{a, b\}$
3. $\{b, a, c, d\}$ _ $\{e, a, b, d\}$

Lack of duplicates/Idempotency

Sets are **idempotent**, which means that duplicates are ignored. So $\{a, b\} = \{a, a, b\} = \{a, b, b, a, b, a, b, a, a\}$. It also holds that $\{a\} = \{a, a\} = \{a, a, a\}$, and so on.

```
import re

def text_to_set(text):
    return set(re.findall(r"\w+", text.lower()))

text1 = "If John slept, then Mary left."
text2 = "If Mary left, then John slept."

set1, set2 = text_to_set(text1), text_to_set(text2)
print("Are the sets identical?")
print(set1 == set2)
```

Example 6 Linguists distinguish between **word types** and **word tokens**. The sentence *dogs love dogs* contain two tokens of the type *dogs*, and one token of the type *love*. The sentences *dogs love* and *dogs love dogs* are different with respect to word tokens, but identical with respect to word types. So if you care about word types rather than word tokens, you're dealing with a set because the only thing that matters is which words the text contains, not how many tokens of each word.

Example 7 Consider the sentence *If police police police, then police police police*. Its set of words (ignoring capitalization) is $\{\text{if}, \text{police}, \text{then}\}$.

Exercise 4 For each one of the following, fill the gap with $=$ or $\neq$ as appropriate:

1. $\{a, b\}$ _ $\{a, a, b, b\}$
2. $\{b, a\}$ _ $\{a, b, a\}$
3. $\{c, b, a, a, d, c\}$ _ $\{a, a, b, d, c, c, c\}$
4. $\{a\}$ _ $\{a, a, a, a, a, a, c, a, a, a, a, a, a\}$

Exercise 5 The sentence *If police police police, then police police police* actually uses two different word types. It just just so happens that both are pronounced and spelled *police*. But one is the noun *police*, the other one the verb *police*. So we might want to annote the string as follows: *If police[N] police[V] police[N], then police[N] police[V] police[N]*. Assume that words are annotated with their part of speech in this fashion. Then what would be the corresponding set of words?

Recap

- Sets are collections of arbitrary objects.
- Sets are unordered and idempotent (= duplicates are ignored).
- Sets can be defined with list notation, e.g. $\{a, b\}$.
- The objects contained in a set are called its *elements* or *members*.
- The symbols $\in$ and $\notin$ are used to indicate membership and non-membership, respectively.
- Occasionally, $\ni$ is used as the mirror image of $\in$.