Functions: Basic notation

Functions, also called **maps** or **mappings**, are ubiquitous in mathematics. The laymen usually thinks of things like $f(x) = x + 1$ when they hear the word *function*, but the concept is much more general.

Functions convert input arguments to an output

Anything can be thought of as a function as long as it takes a fixed number of **arguments** as its input and converts them to an output. Crucially, the output is not allowed to vary while the input is kept the same.

Example 1 A car wash can be regarded as a function that takes as input a car and returns as its output a clean car (in an ideal world, at least). A dirty Dodge Viper comes out as a clean Dode Viper, and a clean Audi A4 still comes out as a clean Audi A4. The output is always perfectly predictable from the input.

Example 2 Suppse $f(x)$ can be randomly chosen between $x + 1$ and $2 \times x$. This is not a function because one and the same input can produce different outputs.

Exercise 1 Let f be a function that takes as its input a number n and returns $n + 1$ on a weekday and $n + 2$ on the weekend.

1. Is f a function?
2. What if f instead takes two arguments: a number n, and the name of the day of the week.

This special property of functions is known as **right uniqueness**. Right uniqueness guarantees that functions are deterministic in the sense that one can predict the output from the input with 100% accuracy.

Caution: The functions used in programming languages are not necessarily functions in the mathematical sense because their output can vary even if the input stays the same.

```
import random
import re

# a programming function that is not a mathematical function
def random_output(number):
    # randomly choose between two outputs
    if random.choice([True, False]):
        return 2 * number
    else:
        return 3 * number

# let's see what happens when we run the function multiple times
for _ in range(10):
    print("The output of random_output({}) is {}".format(5, random_output(5)))
```

Domains and co-domains

Every function has a **domain** and a **co-domain**. The domain is the set of objects from which its arguments can be drawn, and the **co-domain** is the set of objects from which outputs can be drawn. A function is undefined on any arguments that do not belong to its domain.

Example 3 When a car wash is viewed as a function, its domain is the set of all cars (both dirty and clean), whereas the co-domain only contains clean cars.

Exercise 2 What would be the domain and co-domain of a broken car wash that fails to remove even the tiniest speck of dirt?

When defining a function for the first time, it is standard to use the format function-name: domain → co-domain. The actual mapping is specified after that.

Example 4 Let E be the set of English first names. Then the function $f : E \to \{0, 1\}$ maps n to 1 iff n contains at least three syllables.

The mapping from arguments to outputs can be defined in various ways, e.g. in plain English, or as a formula like $f(x) = \frac{(x+x^2+5)^{x+1}}{1000^x}$. For very simple functions whose name was already mentioned, one often writes $x \mapsto y$ instead of $f(x) = y$.

Example 5 Instead of $f(x) = 5 \times x - 3$, we may simple write $x \mapsto 5 \times x - 3$.

Caution: Notice the difference between → and ↦. The first is used when specifying the domain and co-domain, whereas the latter indicates the concrete mapping from an argument to an output.