Functions 2

Composition of Functions

Functions really come in to their own when we think about combining them, using what is called **function composition**. This process allows us to build complicated functions out of much simpler ones.

If we have functions $f\colon A \to B$ and $g\colon B \to C$, then we can define the **composite** of f and g to be a function $g \circ f\colon A \to C$ where $(g \circ f)(x) = g(f(x))$. I.e., apply f first, then apply g.

Function composition is associative, meaning that if we had a further function $h\colon C \to D$, then the following holds:

$$(h \circ g) \circ f = h \circ (g \circ f).$$

We often write composites of more than two functions without the brackets for this reason. E.g., $h \circ g \circ f$.

As an example, the function $s\colon \mathbb{R} \to \mathbb{R}$ defined by

$$s(x) = (e^x + 2)^2$$

can be built up out of other functions.

$$\begin{aligned} f\colon \mathbb{R} &\to \mathbb{R} \\ x &\mapsto e^x \\ g\colon \mathbb{R} &\to \mathbb{R} \\ y &\mapsto y + 2 \\ h\colon \mathbb{R} &\to \mathbb{R} \\ z &\mapsto z^2 \end{aligned}$$

Then if we look at the composite function $h \circ g \circ f$, we have

$$\begin{aligned} (h \circ g \circ f)(x) &= h(g(f(x))) \\ &= h(g(e^x)) \\ &= h(e^x + 2) \\ &= (e^x + 2)^2 \\ &= s(x). \end{aligned}$$

So we can write $s = h \circ g \circ f$. **Video** Visit the URL below to view a video:

`https://www.youtube.com/embed/5G2ynWVyIM8`

Properties of Functions

We will briefly look at some properties which a function may have and give names to these ideas.

Injective

A function is **injective** if it never takes the same value at different inputs from the domain. In other words, a function $f\colon X \to Y$ is injective if

$$f(x) = f(y) \Rightarrow x = y.$$

I.e., if two inputs give the same output, then those inputs must have been the same to start with.

Sometimes injective functions are described as **one-to-one**.

Video Visit the URL below to view a video:
`https://www.youtube.com/embed/BIoVUs0-JtY`

Surjective

A function is **surjective** if every output can be produced by applying the function to some input. In other words, a function $f\colon X \to Y$ is surjective if

$$\forall\, y \in Y \bullet \exists\, x \in X \bullet f(x) = y.$$

I.e., given any potential output $y \in Y$, we can find an input $x \in X$ that actually produces that output, with $f(x) = y$. This is also equivalent to saying that the codomain is equal to the image.

Sometimes surjective functions are described as **onto**.

Video Visit the URL below to view a video:
`https://www.youtube.com/embed/Wp5GhSpdtXA`

Bijective

A function is **bijective**, or a **bijection**, if it is injective and surjective. For a function $f\colon X \to Y$ to be a bijection means that for any given input $x \in X$, there is a unique output $f(x) in Y$. Bijections are incredibly useful since it means we can find an inverse of the function: if we have some $y \in Y$, then since f is a surjection we can find some $x \in X$ such that $f(x) = y$. That f is also an injection means that this is the *only* such x, we know exactly where y came from in X, so we can find an inverse.

Sometimes bijectiones are described as **one-to-one correspondences**.

Recursion

One of the most powerful tools for defining functiosn is the recursive definition. The basic idea of recursion is that the function we are defining is itself used in the body of the definition. It can seem, when you first encounter it, that

it's circular reasoning. However, when used correctly it is perfectly respectable, though we do have to be careful.

The obvious joke here is that if you want to know about recursion you should refer to Section . However, this is actually a *bad* example of recursion, as you never get to the end if you try to follow it through. The key idea for a proper recursive definition is that it must have alternatives, at least one of which does not involve the use of the function itself, and we must eventually use this alternative as we follow the definition through. We will consider recursion again in another light when we encounter mathematical induction.

We will look at some simple examples of recursive functions now.

Factorial

You may have come across the **factorial** function before. We'll call it $\texttt{fact}\colon \mathbb{N} \to \mathbb{N}$. This is usually defined directly one each natural number by saying that

$$\texttt{fact}(n) = n \times (n-1) \times (n-2) \times \ldots \times 3 \times 2 \times 1,$$

with the special case of $\texttt{fact}(0) = 1$ defined separately. For example, $\texttt{fact}(5) = 5 \times 4 \times 3 \times 2 \times 1 = 120$ and $\texttt{fact}(3) = 3 \times 2 \times 1 = 6$. In maths texts, the factorial function is usually denoted by $\texttt{fact}(n) = n!$ instead.

Now we can define this recursively instead, as follows.

$$\texttt{fact}(n) = \begin{cases} 1, & \text{if } n = 0 \\ n \times \texttt{fact}(n-1), & \text{otherwise} \end{cases}$$

Notice that we have the special case for when $n = 0$, which allows the definition to make sense. Otherwise we would just keep multiplying numbers forever. We can check that this makes sense for an example we have already seen:

$$\begin{aligned} \texttt{fact}(5) &= 5 \times \texttt{fact}(4) \\ &= 5 \times 4 \times \texttt{fact}(3) \\ &= 5 \times 4 \times 3 \times \texttt{fact}(2) \\ &= 5 \times 4 \times 3 \times 2 \times \texttt{fact}(1) \\ &= 5 \times 4 \times 3 \times 2 \times 1 \times \texttt{fact}(0) \\ &= 5 \times 4 \times 3 \times 2 \times 1 \times 1 \\ &= 120. \end{aligned}$$

Fibonacci Numbers

The Fibonacci numbers are a sequence which begins 0, 1, 1, 2, 3, 5, ..., and continues with each number being the sum of the previous two. More formally, we have a function $\texttt{fib}\colon \mathbb{N} \to \mathbb{N}$ defined recursively as follows.

$$\texttt{fib}(n) = \begin{cases} 0, & \text{if } n = 0 \\ 1, & \text{if } n = 1 \\ \texttt{fib}(n-1) + \texttt{fib}(n-2), & \text{otherwise} \end{cases}$$

The Fibonacci numbers are interesting mathematically and have applications in many areas, including modelling the growth of plants and animals.

Concept Checks

Test Yourself Visit the URL below to try a numbas exam:
`https://numbas.mathcentre.ac.uk/question/100693/evaluating-composite-functions/embed`