4.12.1 Functional Programming

Basics of Functional Programming

Functional programming is a programming paradigm based on the use of functions to process data. It is a type of declarative programming.

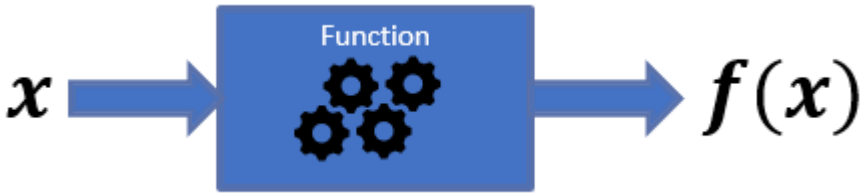
Programs are constructed by combining functions that take inputs and return outputs.

Functions are **first-class objects**, meaning they can be passed as arguments to other functions and returned as results.

Examples of functional programming languages include Haskell, Lisp and Erlang. Languages such as Python, Perl and C# also support functional programming.

The **domain** is a set from which a function's input values are chosen. A **co-domain** is a set of the possible outputs that can be a result of applying the function to the values in the domain.

A function takes inputs, performs calculations and outputs a result



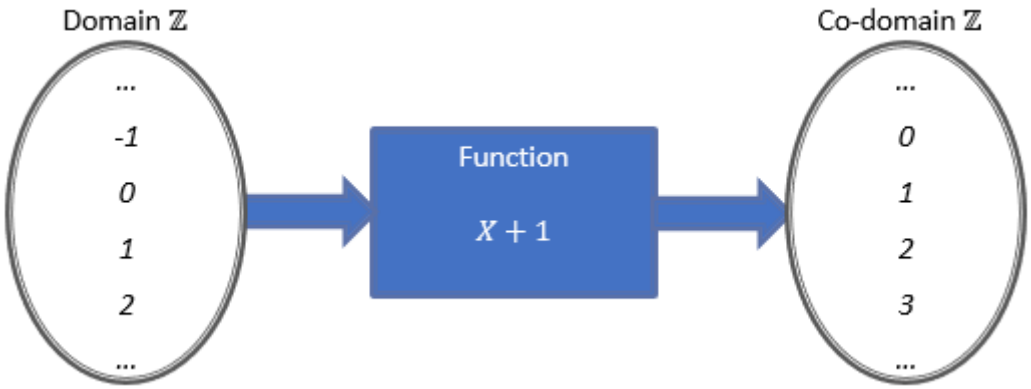
Example of a function

Let $f(x) = x + 1 \qquad f: \mathbb{Z} \longrightarrow \mathbb{Z}$

The function f takes an input x, adds 1 to it and outputs the result.

The domain is the set of integers $\mathbb{Z}$ and the co-domain is the set of integers $\mathbb{Z}$.

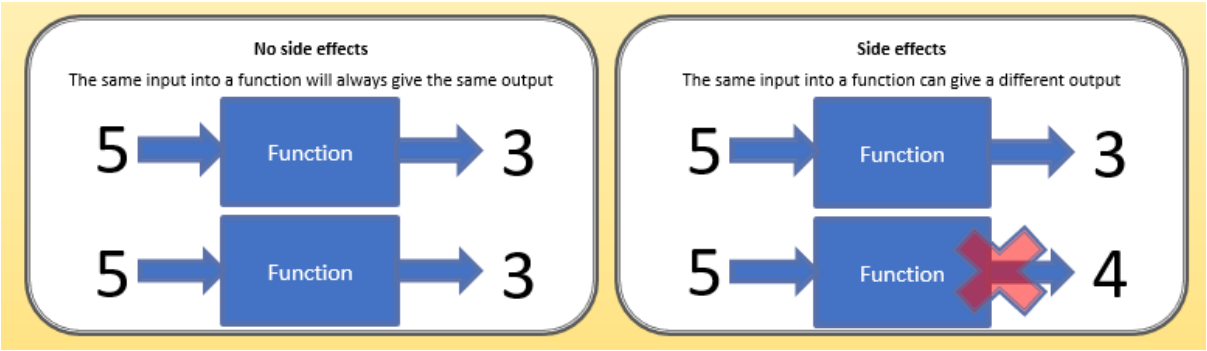
$$\dots$$
$$f(-1) = 0$$
$$f(0) = 1$$
$$f(1) = 2$$
$$f(2) = 3$$
$$\dots$$



Advantages of Functional Programming

- **Immutable data structures** - Once a value has been assigned, it cannot be changed.
- **Statelessness** – Functions outputs only depend on their inputs, so there can be **no side effects.** This makes it easier to predict how a program will behave and makes programs easier to predict, test and debug.
- **Order of execution is not fixed –** The program code does not determine the order in which functions are executed; this can be determined at run-time.

This makes functional programming a great fit for **parallel processing** allowing the processing to be split over many cores or machines.



Partial Function Application

Function application is the process of giving particular inputs to a function.

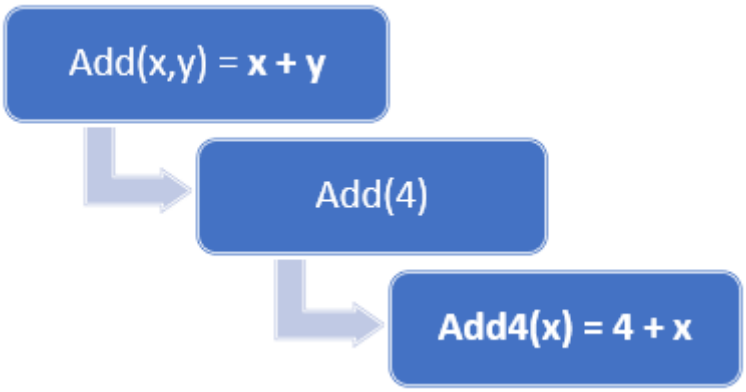
Partial function application is the process of applying a function to **one** of its arguments to output a new function.

Let the function add be defined as such:

$$add(a, b) = a + b \qquad add : \mathbb{Z} \to \mathbb{Z} \to \mathbb{Z}$$

A partial application of the add function with the argument 4.

1. The argument 4 is supplied to the function add.
2. This produces a new function $add4$ with one less argument.



Composition

Function composition combines two or more functions so that the output of one function becomes the input of the next.

$$f(x) = 4 * x$$

$$g(x) = x + 2$$

$$f \circ g(x) = 4 * (x + 2)$$

The function g is applied to x. Next function f is applied to the result of $g(x)$.

The co-domain of g must match the domain of f so that the functions can be composed.