4.12.2 Writing Functional Programs

List Operations

Functional programs commonly use operations that manipulate lists.

Operation	*Description*	*AQA Functional Code*	*Result*
Construct a list	Creates a new list	*listA = [1, 2, 3, 4]* *listB = [ ]*	
Return head of a list	Returns the first item in a list	*head(listA)*	1
Return tail of a list	Returns a list of all items except the first item	*tail(listA)*	[2, 3, 4]
Prepend an item onto a list	Insert a new item to the front of a list	5 : listA	[5, 1, 2, 3, 4]
Append an item onto a list	Insert a new item to the end of a list	listA : 5	[1, 2, 3, 4, 5]

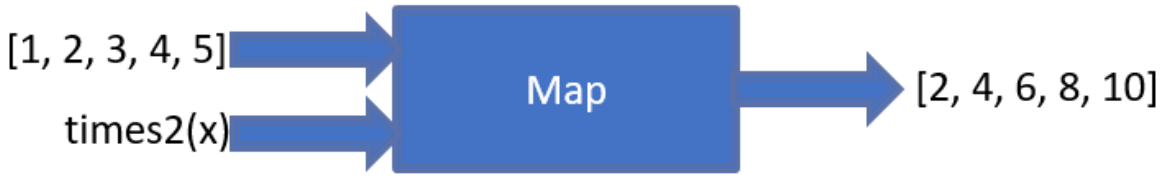
Higher Order Functions

A **higher order function** is a function that can:

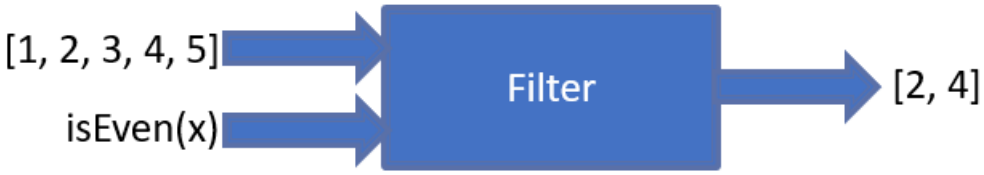
- Take a function as an argument
- Return a function as a result
- … or both.

Examples of higher order functions are **map, filter and reduce**.

- **Map** – takes two arguments; a list and a function, and outputs the result of applying that function to each item in the list.



- **Filter** – takes two arguments; a list and a function that returns a Boolean for a condition, and outputs a list of items that satisfy that condition.



- **Reduce/fold** – takes three arguments; a list, a start value and a function and returns a single value which is the result of recursively applying that function to each item in the list.

Map

Map – takes two arguments; a list and a function, and outputs the result of applying that function to each item in the list.

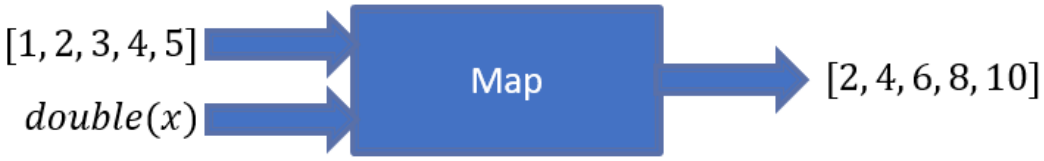
Below is a definition for a list $temp$ and a function $double$.

$$temps = [1,\ 2,\ 3,\ 4,\ 5]$$

$$double\ a\ = a * a$$

The results of calling the higher-order function map with $temp$ and $double$ as arguments.

$$map\ double\ temps\ \rightarrow [2,\ 4,\ 6,\ 8,\ 10]$$



Filter

Filter – takes two arguments; a list and a function that returns a Boolean for a condition, and outputs a list of items that satisfy that condition.

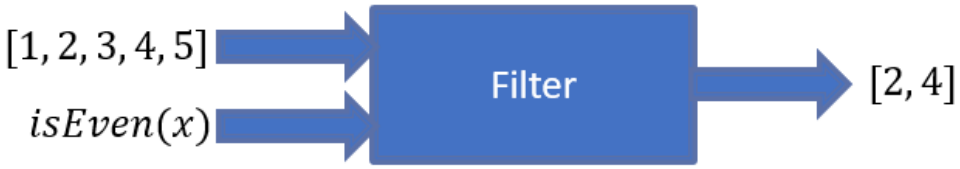
Below is a definition for a list $temp$ and a function $isEven$.

$$temps = [1,\ 2,\ 3,\ 4,\ 5]$$

$$isEven\ a\ = a\ mod\ 2 == 0$$

The results of calling the higher-order function map with $temp$ and $isEven$ as arguments.

$$filter\ isEven\ temps\ \rightarrow [2,\ 4]$$



Reduce

Reduce/fold – takes three arguments; a list, a start value and a function and returns a single value which is the result of recursively applying that function to each item in the list.

Below is a definition for a list $temp$ and a function add.

$$temps = [1,\ 2,\ 3,\ 4,\ 5]$$

$$add\ (a, b) = a + b$$

The results of calling the higher-order function map with $temp$, 0 and add as arguments.

$$reduce\ add\ 0\ temps\ \rightarrow 15$$

Map – A recursive example

Some function definitions will involve recursion.

Below is a definition for a list $temp$, a function $double$ and a recursive definition for map.

$$double\ a\ = a * a$$

$$map\ f\ [\,] = []$$

$$map\ f\ (x{:}\,xs) = fx : map\ f\ xs$$

In the map definition, x is the head of the list argument and xs is the tail.

$$\text{map}\ double\ [1,\ 2,\ 3] \rightarrow 1 : \text{map}\ double\ [2,\ 3]$$

$$\text{map}\ double\ [2,\ 3] \rightarrow 4 : \text{map}\ double\ [3]$$

$$\text{map}\ double\ [3] \rightarrow 6 : \text{map}\ double\ [\,]$$

$$\text{map}\ double\ [\,] = [\,]$$

$$1 : 4 : 6 : [\,] = [1,\ 4,\ 6]$$