# Learning by Example: Summation

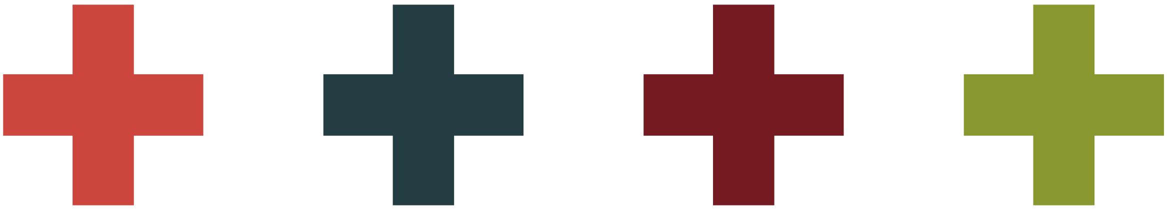
In this lesson, we will write a summation program using the methods we have learned so far.

**We'll cover the following**

- 1. Sum of integers between ‘a’ and ‘b’
- 2. Sum of the cubes of all the integers between ‘a’ and ‘b’
- 3. Sum of the factorials of all the integers between ‘a’ and ‘b’

Let’s look at an example where we will learn how to implement higher-order functions and also understand the type of scenarios in which they should be used.

Below, we are going to define a set of recursive functions which all perform some sort of summation.



## 1. Sum of integers between ‘a’ and ‘b’ #

This code requires the following environment variables to execute:

| | |
|---|---|
| LANG | C.UTF-8 |

```
def sumOfInts(a: Int, b: Int): Int ={
  if(a>b) 0
  else a + sumOfInts(a+1,b)
}

println(sumOfInts(1,5))
```

## 2. Sum of the cubes of all the integers between ‘a’ and ‘b’ #

This code requires the following environment variables to execute:

This code requires the following environment variables to execute:

| | |
|---|---|
| LANG | C.UTF-8 |

```
def cube(x: Int): Int ={
  x * x * x
}

def sumOfCubes(a: Int, b: Int): Int ={
  if(a>b) 0
  else cube(a) + sumOfCubes(a+1,b)
}

println(sumOfCubes(1,5))
```

## 3. Sum of the factorials of all the integers between ‘a’ and ‘b’

```
def factorial(x: Int): Int = {
  if(x==0) 1
  else x * factorial(x-1)
}

def sumOfFactorials(a: Int, b: Int): Int ={
  if(a>b) 0
  else factorial(a) + sumOfFactorials(a+1,b)
}

println(sumOfFactorials(1,5))
```

You might have noticed that all the functions we have defined above are different cases of:

$$\sum_{n=a}^{b} f(n)$$

With $f$ being any of the functions above.

Can we factor out the common pattern into a single function rather than have three separate functions?

---

In the next lesson, we will try to solve the above problem using higher-order functions.