Higher-Order Functions

Functions

- Functional languages treat functions as first-class values => a function can be passed as a parameter and returned as a result => flexible way to compose programs
- Functions that take other functions as parameters or that return functions as results are called *higher order functions*
- The type A => B is the type of a function that takes an argument of type A and returns a result of type B

Anonymous Functions

- Passing functions as parameters leads to the creation of many small functions and sometimes it becomes tedious to have to define and name all these functions
- Function literals are called anonymous functions and let us write a function without giving it a name
- The type of the parameter can be omitted if it can be inferred by the compiler from the context
- An anonymous function can always be expressed using def => anonymous functions are syntactic sugar

Exercise: *Write a function that returns* $\sum_{x=a}^{b} f(x)$*, where a, b, f are given as function parameters. Try writing a tail-recursive version of this function.*

```
def sum(f: Int => Int, a: Int, b: Int): Int = {
  def loop(a: Int, acc: Int): Int =
    if (a > b)
      acc
    else
      loop(a + 1, acc + f(a))

  loop(a, 0)
}
```