Blocks and Lexical Scope

Nested Functions

- It's good functional programming style to split up a task into small functions
- This creates some functions' names that matter only for the implementation of a more important function and normally we would not like to access these functions directly
- We can achieve this and at the same time avoid "name-space pollution" by putting the auxiliary functions inside the more important function

Blocks in Scala

- A block is delimited by braces (in Scala 3, they are optional): { ... }
- It contains a sequence of definitions or expressions
- The last element of the block is an expression that defines its value
- This return expression can be preceded by auxiliary definitions
- Blocks are themselves expressions; a block may appear everywhere an expression can
- The definitions inside a block are only visible from within the block and they shadow definitions of the same names outside the block
- Definitions of outer blocks are visible inside a block unless they are shadowed.

Question: *What is the value of* ***result*** *in the following program?*

```
val x = 0
def f (y: Int) = y + 1
val y =
  val x = f(3)
  x * x
val result = y + x
```

Answer: 16

Semicolons

- If there are more than one statements on a line, they need to be separated by semicolons
- Semicolons at the end of lines are usually left out