You know from the last module that the call-by-name and call-by-value evaluation strategies reduce an expression to the same value, as long as both evaluations terminate.

But what if termination is not guaranteed?

We have:

- If CBV evaluation of an expression $e$ terminates, then CBN evaluation of $e$ terminates, too.
- The other direction is not true.
Non-termination example

Question: Find an expression that terminates under CBN but not under CBV.
Let's define

```python
def first(x: Int, y: Int) = x
```

and consider the expression `first(1, loop)`.

**Under CBN:**

```
first(1, loop)
```

**Under CBV:**

```
first(1, loop)
```
Scala’s evaluation strategy

Scala normally uses call-by-value.

But if the type of a function parameter starts with => it uses call-by-name.

Example:

```scala
def constOne(x: Int, y: => Int) = 1
```

Let’s trace the evaluations of

```scala
constOne(1+2, loop)
```

and

```scala
constOne(loop, 1+2)
```
Trace of constOne(1 + 2, loop)

constOne(1 + 2, loop)
Trace of \texttt{constOne(loop, 1 + 2)}

\texttt{constOne(loop, 1 + 2)}