Tutorial Worksheet 1

Duncan Paul Attard · August 1, 2017

The document was typeset using *XeTeX*

Task 1 · Installing Erlang

The pre-built Erlang package for your operating system can be downloaded from Erlang Solutions.

Task 2 · Testing the Erlang shell

To confirm that Erlang has been successfully installed, we can try to invoke its shell. This can be done by typing `erl` in your operating system's terminal. You should get something that looks similar to the following:

```
duncan@term:/example$ erl

Erlang/OTP 18 [erts-7.2] [source] [smp:4:4] [async-threads:10] [kernel-poll:false]
Eshell V7.2  (abort with ^G)
1> _
```

Type `1 + 1.` followed by enter. What happens? Try out other arithmetic expressions for yourself. When ready, quit the Erlang shell by typing `q().`

Shell expressions

Erlang expressions entered in the shell *must* terminate with the period symbol '`.`'. Issuing enter without typing in '`.`' gives us the opportunity to type in code that can extend over multiple lines (*e.g.* a lengthy function). Code spanning multiple lines can then be terminated accordingly by typing '`.`'.

To recall the latest executed shell command, tap the *up* arrow key on your keyboard.

Task 3* · Fibonacci numbers

Write a recursive function `fib(N)` that calculates the fibonacci number for a given non-negative number (*i.e.*, $n \geq 0$). Try to run your function using very large numbers and observe what happens.

Task 4* · Fibonacci numbers revisited

Why do you think the `fib(N)` did not perform as well as expected when tested with large numbers?

Implement a new tail-recursive function `fib2(N)` that calculates the fibonacci number using *two* accumulator values to store the values for $n-1$ and $n-2$. The following snippet should act as a scaffolding for your work.

```
fib2(N) -> fib2(N, 1, 0).
fib2(0, Nminus1, Nminus2) ->
  % Base case implementation.
  ok;
fib2(N, Nminus1, Nminus2) ->
  % Recursive case implementation.
  ok.
```