A tour of Nix

12 / 35 Partial application

prev next

Partial application: A function returning another function that might return another function, but each returned function can take several parameters.

Let's have a look into `functions` and how they are defined and called:

- solve each question: `ex00, ex01, ...` by only modifying X with a `value` of type `Int`

Note: If you see `ex03 = <LAMBDA>;` after `run`, this means that `ex03` is bound to a `function`.

Note: See video [@youtube]

```
let
  b = 1;
  fu0 = (x: x);
  fu1 = (x: y: x + y) 4;
  fu2 = (x: y: (2 * x) + y);
in
rec {
  ex00 = fu0 X;      # must return 4
#  ex01 = (fu1) X;    # must return 5
#  ex02 = (fu2 X ) X; # must return 7
#  ex03 = (fu2 X );    # must return <LAMBDA>s
#  ex04 = ex03 X;     # must return 7
#  ex05 = (n: x: (fu2 x n)) X X; # must return 7
}

```

reset solution run

```
let
  b = 1;
  fu0 = (x: x);
  fu1 = (x: y: x + y) 4;
  fu2 = (x: y: (2 * x) + y);
in
rec {
  ex00 = fu0 4;      # must return 4
  ex01 = (fu1) 1;    # must return 5
  ex02 = (fu2 2) 3; # must return 7
  ex03 = (fu2 3);    # must return <LAMBDA>
  ex04 = ex03 1;     # must return 7
  ex05 = (n: x: (fu2 x n)) 3 2; # must return 7
}
```