Functions that create lists

- Let us now define a function that outputs a list
  - We will use both pattern matching and recursion, as before, but this time the output will also be a list
  - We will define the Append function

The simple Append function is tail recursive

- We will see this by translating Append into the kernel language of the functional paradigm
- This translation shows that the recursive call is last
- This works because of single assignment: we create the output list before doing the recursive call

The kernel language

- As we mentioned in lesson 1, the kernel language is the simple core language of a programming paradigm
  - We have now seen enough concepts to introduce the kernel language of the functional paradigm

- All programs in the functional paradigm can be translated into the kernel language
  - $\boxed{\text{All intermediate results of calculations are visible with identifiers}}$ Kernel principle
  - All functions become procedures with one extra argument
  - Nested function calls are unnested by introducing new identifiers

- The kernel language is the first part of the formal semantics of a programming language
  - The second part is the *abstract machine* seen in lesson 6

Kernel language of the functional paradigm

- <s> ::= **skip**
  | $<s>_1$ $<s>_2$
  | **local** <x> **in** <s> **end**
  | $<x>_1$=$<x>_2$
  | <x>=<v>
  | **if** <x> **then** $<s>_1$ **else** $<s>_2$ **end**
  | **proc** {<x> $<x>_1$ … $<x>_n$} <s> **end**
  | {<x> $<y>_1$ … $<y>_n$}
  | **case** <x> **of** <p> **then** $<s>_1$ **else** $<s>_2$ **end**
- <v> ::= <number> | <list> | …
- <number> ::= <int> | <float>
- <list>, <p> ::= nil | <x> | <x> '|' <list>

Almost complete!

We will see the full kernel language in lesson 4