Pattern matching

- List patterns in the examples earlier.
- For example, the reverse function:

In Haskell:

```
reverse        :: [a] → [a]
reverse []      = []
reverse (x:xs) = reverse xs ++ [x]
```

In Scala:

```
def reverse [T] (xs : List[T]) : List[T] =
    xs match {
       case Nil => Nil
       case x::ys => reverse(ys) ::: List(x)
    }
```

(*)

- Other patterns such as pair-patterns:

  In Haskell:

```
True && True = True
_    && _    = False
```

  In Scala:

```
def AND (x:Boolean,y:Boolean) : Boolean =
    (x,y) match {
        case (true,true) => true
        case (_,_) => false
    }
```

Anonymous functions

- Consider the following function in Haskell:

```
odds n = map (λx → x*2 + 1) [0..n-1]
```

- In Scala, this can be done as

```
def odds (n:Int) =
    List.range(0,n-1) map ((x:Int) => x*2+1)
```

Note, besides the anonymous function, the use of List.range(m,n) (in Haskell, this is [m,n-1]).