Handy-Dandy Function Prototypes

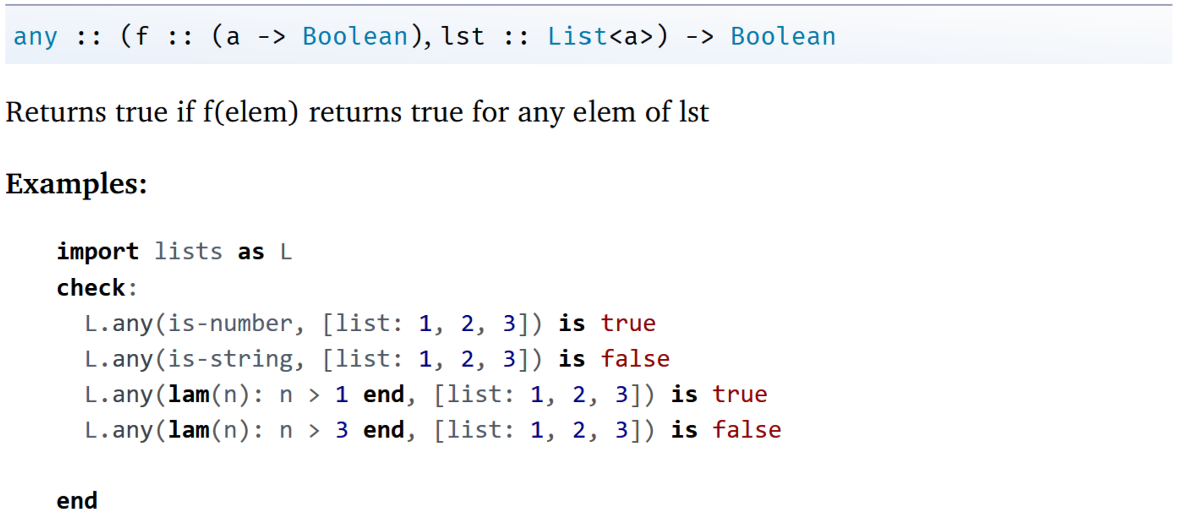
CMPU 101 – Problem Solving and Abstraction

Peter Lemieszewski

Various Function Prototypes

- We’ve seen these before
- And, they are better served as specific examples from your
  - Homework assignments,
  - Labs
- Their usage is documented in previous lecture notes
- This is not an inclusive list, but a way to help you sort through your note taking.

Compare to my-any



```
any :: (f :: (a -> Boolean), lst :: List<a>) -> Boolean
```

Returns true if f(elem) returns true for any elem of lst

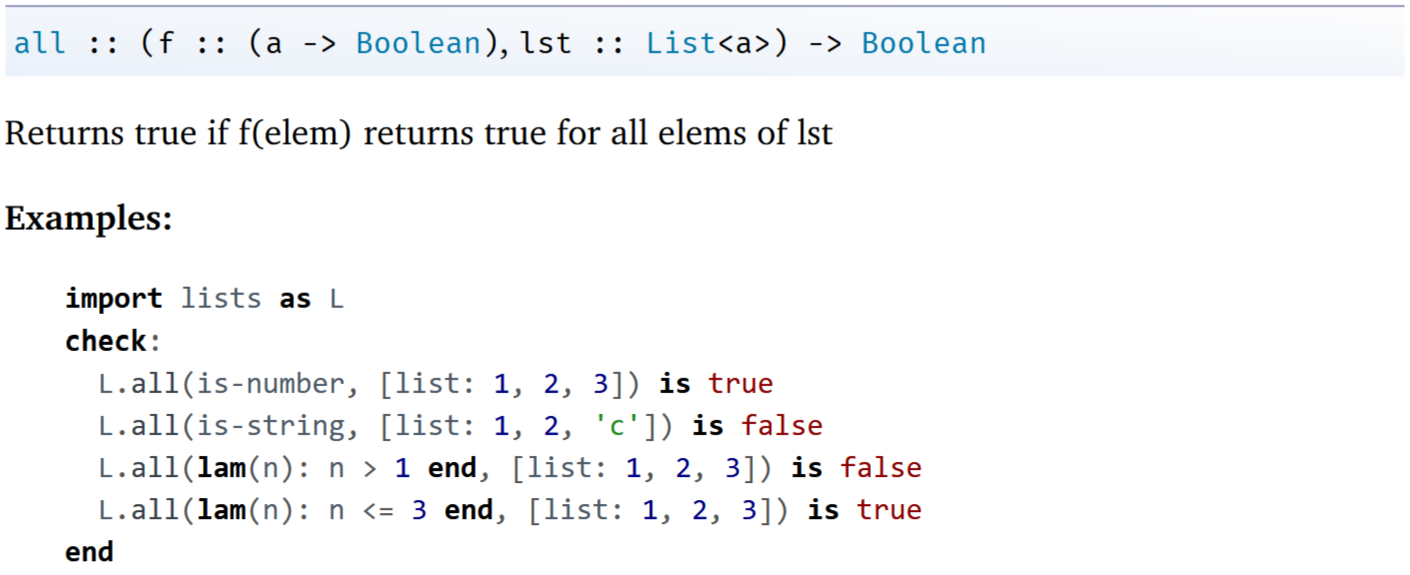
Examples:

```
import lists as L
check:
  L.any(is-number, [list: 1, 2, 3]) is true
  L.any(is-string, [list: 1, 2, 3]) is false
  L.any(lam(n): n > 1 end, [list: 1, 2, 3]) is true
  L.any(lam(n): n > 3 end, [list: 1, 2, 3]) is false

end
```

- Note: this is directly from pyret documentation.
- However, it is not necessary to:
    - *"Import lists as L"* and
    - use "*L.*" to access list functions.
- This is true for subsequent functions here too.

Compare to my-all



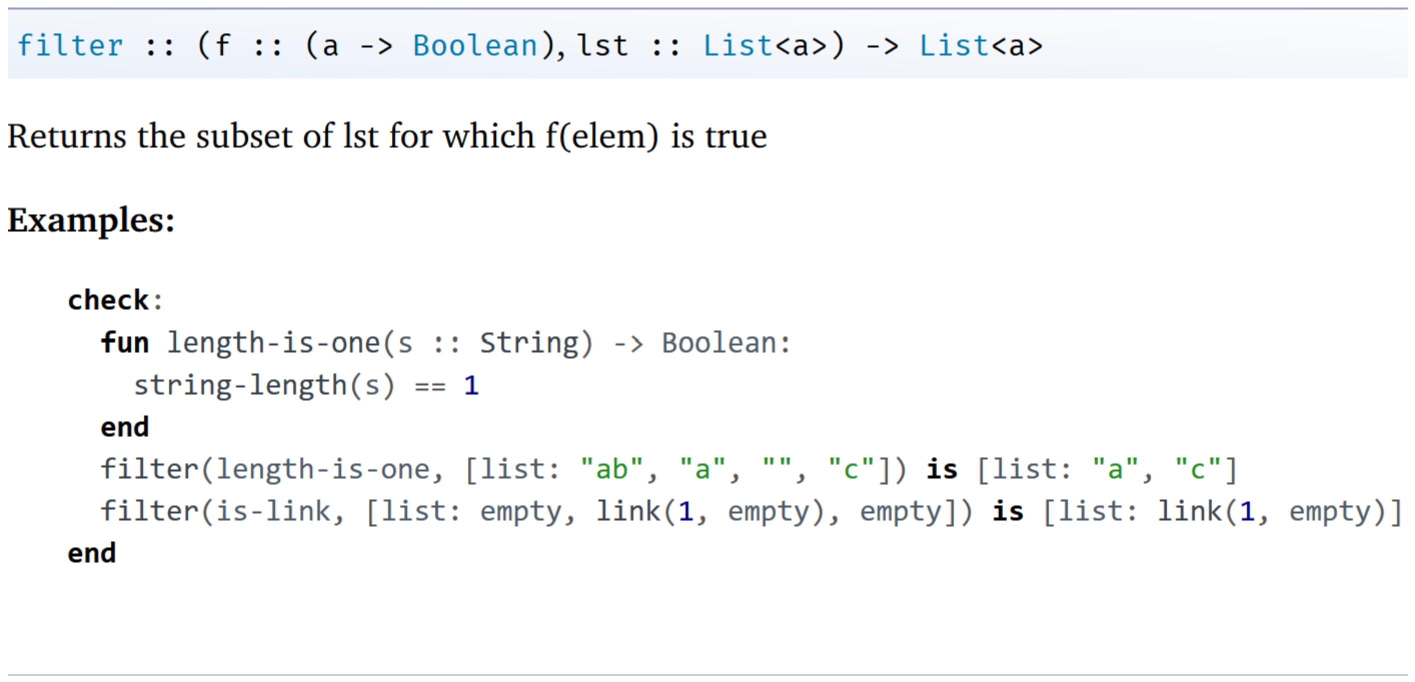
```
all :: (f :: (a -> Boolean), lst :: List<a>) -> Boolean
```

Returns true if f(elem) returns true for all elems of lst

Examples:

```
import lists as L
check:
  L.all(is-number, [list: 1, 2, 3]) is true
  L.all(is-string, [list: 1, 2, 'c']) is false
  L.all(lam(n): n > 1 end, [list: 1, 2, 3]) is false
  L.all(lam(n): n <= 3 end, [list: 1, 2, 3]) is true
end
```

Recall A Previous Homework Assignment



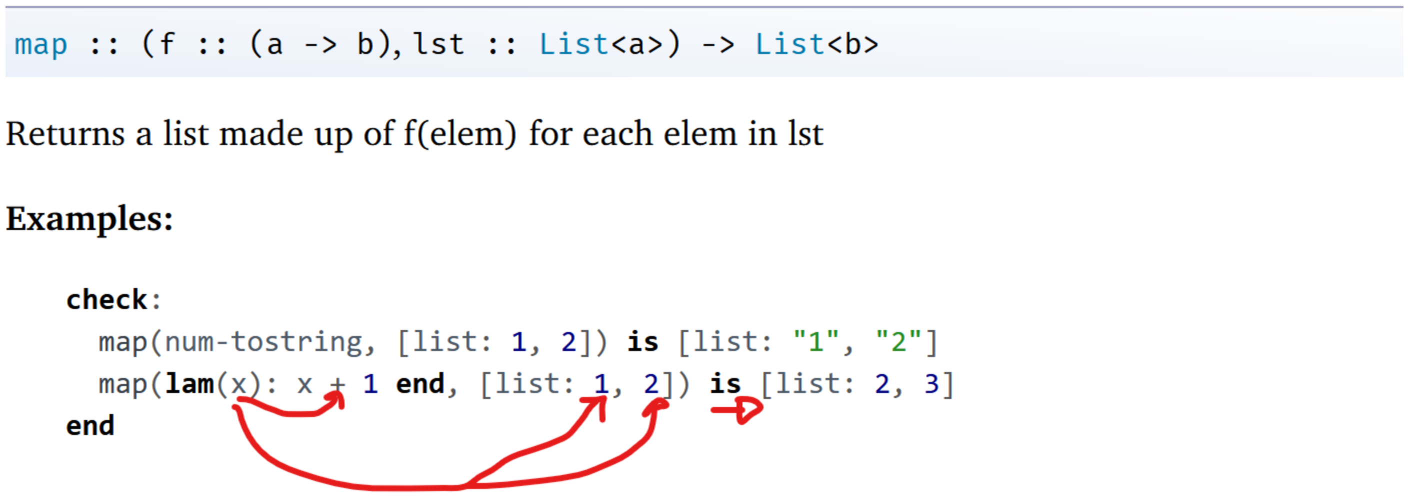
```
filter :: (f :: (a -> Boolean), lst :: List<a>) -> List<a>
```

Returns the subset of lst for which f(elem) is true

Examples:

```
check:
  fun length-is-one(s :: String) -> Boolean:
    string-length(s) == 1
  end
  filter(length-is-one, [list: "ab", "a", "", "c"]) is [list: "a", "c"]
  filter(is-link, [list: empty, link(1, empty), empty]) is [list: link(1, empty)]
end
```

Recall how map was used in homework #5



```
map :: (f :: (a -> b), lst :: List<a>) -> List<b>
```

Returns a list made up of f(elem) for each elem in lst

Examples:

```
check:
  map(num-tostring, [list: 1, 2]) is [list: "1", "2"]
  map(lam(x): x + 1 end, [list: 1, 2]) is [list: 2, 3]
end
```