S5: Data Abstraction

CS1101S AY20/21 Sem 1

Studio 2D

Lee Wei Min

Mastery Check

- One team has already completed
- Please book by this week

Points

```
/* Construct: */
const p = make_point(0.5, 0.25);

/* Select: */
const x = x_of(p);
const y = y_of(p);
```

```
function make_point(x, y) {
    return dimension =>
        dimension === "one" ? x : y;
}
function x_of(p) {
    return p("one");
}
function y_of(p) {
    return p("two");
}
```

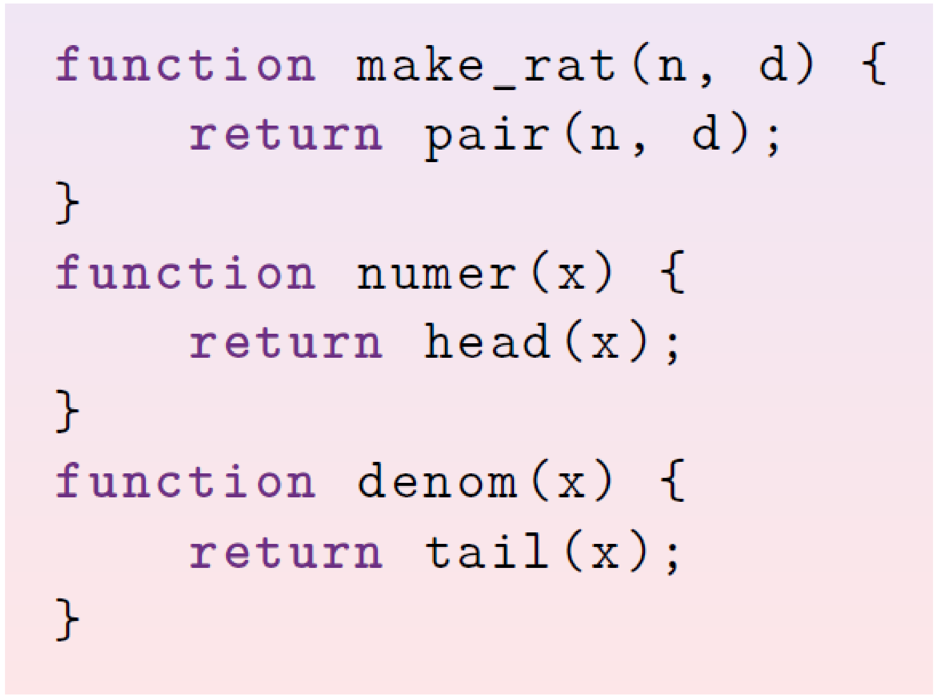
Functional Expressionism

```
const pair = (x, y) => f => f(x, y);


const head = p => p((x, y) => x);

const tail = p => p((x, y) => y);
```

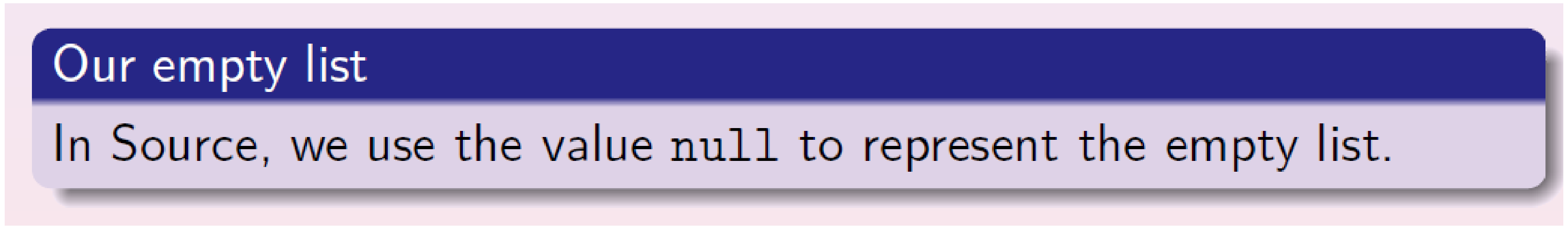
Rational Numbers



```
function make_rat(n, d) {
    return pair(n, d);
}
function numer(x) {
    return head(x);
}
function denom(x) {
    return tail(x);
}
```

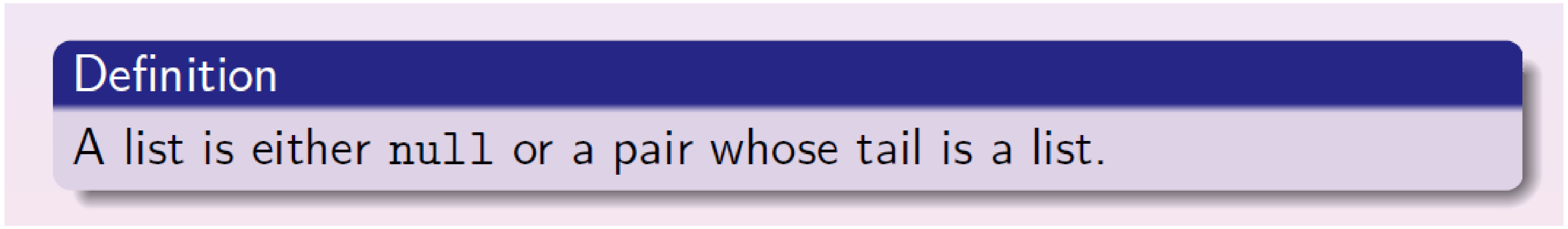
```
function add_rat(x, y) {
    return make_rat(numer(x) * denom(y) +
                    numer(y) * denom(x),
                    denom(x) * denom(y));
}
function sub_rat(x, y) {
    return make_rat(numer(x) * denom(y) -
                    numer(y) * denom(x),
                    denom(x) * denom(y));
}
```

List



Our empty list

In Source, we use the value `null` to represent the empty list.



Definition

A list is either `null` or a pair whose tail is a list.

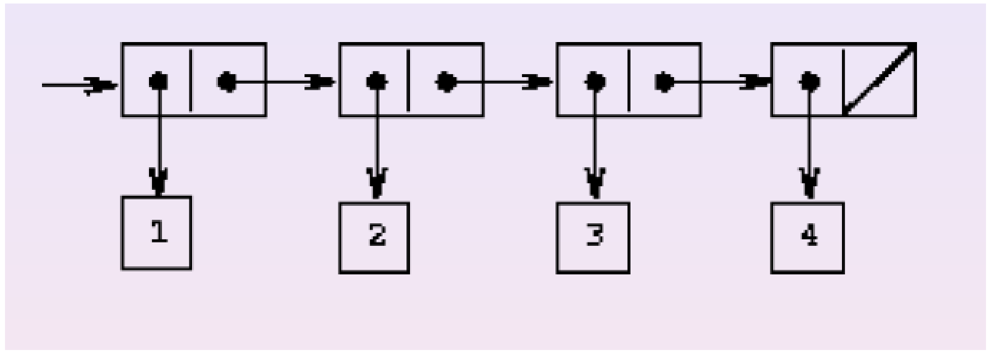
- `pair(x, y)`: returns pair made of `x` and `y`
- `is_pair(p)`: returns `true` iff `p` is a pair
- `null`: represents an empty list
- `is_null(xs)`: returns `true` iff `xs` is the empty list `null`
- `head(p)`: returns the head (first component) of the pair `p`
- `tail(p)`: returns the tail (second component) of the pair `p`
- `list(x1,...,xn)`: returns list whose first element is `x1`, second element is `x2`, etc. and last element is `xn`

Example

```
pair(1, pair(2, pair(3, null)));
```

is printed as

```
[1, [2, [3, null]]]
```



Definition

The length of the empty list is 0, and the length of a non-empty list is one more than the length of its tail.

```
function length(xs) {
    return is_null(xs)
           ? 0
           : 1 + length(tail(xs));
}
```

```
function length_iter(xs) {
    function len(ys, counted_so_far) {
        return is_null(ys)
               ? counted_so_far
               : len(tail(ys),
                     counted_so_far + 1);
    }
    return len(xs, 0);
}
```

Studio