CSE 115
Introduction to Computer Science I
Road map

Review exercises
key-value maps in JavaScript

objects vs. maps

- An object is a key-value map
  - Some downsides (e.g. not anything can be a key)
  - Some advantages (e.g. simplicity, direct JSON support)

- A map is a key-value map

  - Some downsides (no direct JSON support)
  - Some advantages (e.g. anything can be a key, richer operations)

We will explore objects as key-value maps for now.
Operations

Create
> x = {}
> x = { 'a': 1, 'b': 2, 'c': 3, 'd': 4 }

Add/Update
> x['c'] = 12
> x['b'] = 7
> x['z'] = 3

Retrieve
> x['c']
> x.c

Remove
> delete x['c']
> delete x.c

Membership
>>> 'c' in x
>>> ! ('c' in x)
Operations

**Keys**
> `Object.keys(x)`

**Values**
> `Object.values(x)`

**Pairs**
>

This is not built-in as in Python.

No problem - we can define our own function!

```javascript
function items(map) {
  var answer = [];
  for (var k in map) {
    var p = {};
    p[k] = map[k];
    answer.push(p);
  }
  return answer;
}
```
Operations

**Keys**

>`Object.keys(x)`

**Values**

>`Object.values(x)`

**Pairs**

>`Object.entries(x)`

Whoops! Dr. Alphonce goofed. `Object.entries(x)` will do the job.
Road map

Review

▶ exercises ◀
A shopping carts dictionary pairs customer names with a list of product names they intend to purchase, as in:

```python
shoppingCarts = {
    'joe': ['milk', 'cookies', 'spinach'],
    'amy': ['carrots', 'flour', 'sugar', 'milk', 'cereal']
}
```

A price list dictionary pairs product names with prices, as in:

```python
priceList = {
    'milk': 1.49, 'cookies': 2.00, 'spinach': 0.49,
    'carrots': 1.00, 'flour': 2.49, 'sugar': 2.29,
    'cereal': 1.79
}
```

Define a function named `cartTotals` that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for the items in their respective shopping carts.
exercise 1

A shopping carts dictionary pairs customer names with a list of product names they intend to purchase, as in:

```
shoppingCarts = {
    'joe' : [ 'milk', 'cookies', 'spinach' ],
    'amy' : ['carrots', 'flour', 'sugar', 'milk', 'cereal' ]
}
```

A price list dictionary pairs product names with prices, as in:

```
priceList = { 'milk' : 1.49, 'cookies' : 2.00, 'spinach' : 0.49,
              'carrots' : 1.00, 'flour' : 2.49, 'sugar' : 2.29, ...
}
```

Define a function named `cartTotals` that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for the items in their respective shopping carts.
exercise 1

A shopping carts dictionary pairs customer names with a list of product names they intend to purchase, as in:

```
shoppingCarts = {
    'joe' : ['milk', 'cookies', 'spinach'],
    'amy' : ['carrots', 'flour', 'sugar', 'milk', 'cereal']
}
```

A price list dictionary pairs product names with prices, as in:

```
priceList = {
    'milk' : 1.49, 'cookies' : 2.00, 'spinach' : 0.49,
    'carrots' : 1.00, 'flour' : 2.49, 'sugar' : 2.29, 'cereal' : 1.79
}
```

Define a function named cartTotals that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for the items in their respective shopping carts.
exercise 1

A shopping carts dictionary pairs customer names with a list of product names they intend to purchase, as in:

```
shoppingCarts = {
    'joe' : [ 'milk', 'cookies', 'spinach' ],
    'amy' : [ 'carrots', 'flour', 'sugar', 'milk', 'cereal' ]
}
```

A price list dictionary pairs product names with prices, as in:

```
priceList = { 'milk' : 1.49, 'cookies' : 2.00, 'spinach' : 0.49,
              'carrots' : 1.00, 'flour' : 2.49, 'sugar' : 2.29,
              'cereal' : 1.79 }
```

Define a function named `cartTotals` that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for the items in their respective shopping carts.
exercise 1

Problem-solving approach: decompose problem into subproblems that are easier to solve. Combine solutions to subproblems into solution to original problem.

Question: what are subproblems?
A first decomposition step...

// to build the dictionary of customers and total costs

... for each customer in carts ... 
... compute the total for their cart ... 
... add customer : total pair to the answer ...
...decomposing further...

// to build the dictionary of customers and total costs

    ... for each customer in carts ...
    ... compute the total for their cart ...
    ... add customer : total pair to the answer ...

// to compute the total for a given customer's cart

    ... look up the customer's cart ...
    ... compute the total for that cart ...
...and a little more.

// to build the dictionary of customers and total costs

... for each customer in carts ...
... compute the total for their cart ...
... add customer : total pair to the answer ...

// to compute the total for a given customer's cart

... look up the customer's cart ...
... compute the total for that cart ...

// to compute the total for a single cart

... for each item in cart ...
... look up the cost of that item in prices ...
... add the cost to total ...
Tackle the simplest problem first…

// to compute the total for a single cart

... for each item in cart ...
... look up the cost of that item in prices ...
... add the cost to total ...
A shopping cart is an array of product names, as in:

```
cart = [ 'milk', 'cookies', 'spinach' ]
```

A price list dictionary pairs product names with prices, as in:

```
priceList = { 'milk' : 1.49, 'cookies' : 2.00, 'spinach' : 0.49,
             'carrots' : 1.00, 'flour' : 2.49, 'sugar' : 2.29,
             'cereal' : 1.79 }
```

Define a function named cartTotal that takes a shopping cart and a price list dictionary, and returns the total amount for the items in that cart.
This follows a familiar pattern...

```javascript
// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for (var item of cart) {
        . . . look up the cost of that item in prices . . .
        . . . add the cost to total . . .
    }
    return total;
}
```
...spell out the details of the loop...

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for (var item of cart) {
        var price = prices[item];
        ... add the cost to total ...
    }
    return total;
}
...spell out the details of the loop...

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for (var item of cart) {
        var price = prices[item];
        total = total + price;
    }
    return total;
}
We've solved part of the overall problem!

// to build the dictionary of customers and total costs

... for each customer in carts ...
... compute the total for their cart ...
... add customer : total pair to the answer ...

// to compute the total for a given customer's cart

... look up the customer's cart ...
... compute the total for that cart ...

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for (var item of cart) {
        var price = prices[item];
        total = total + price;
    }
    return total;
}
Tackle this problem second

```javascript
// to compute the total for a given customer's cart
.
.
.
look up the customer's cart
.
.
compute the total for that cart
.
.
// to compute the total for a single cart
function singleCartTotal(cart, prices) {
  var total = 0;
  for (var item of cart) {
    var price = prices[item];
    total = total + price;
  }
  return total;
}
```
exercise 1(b)

Assume you have completed exercise 1(a).

A shopping carts dictionary pairs customer names with a list of product names they intend to purchase, as in:

```python
shoppingCarts = {
    'joe': ['milk', 'cookies', 'spinach'],
    'amy': ['carrots', 'flour', 'sugar', 'milk', 'cereal']
}
```

A price list dictionary pairs product names with prices, as in:

```python
priceList = {
    'milk': 1.49, 'cookies': 2.00, 'spinach': 0.49,
    'carrots': 1.00, 'flour': 2.49, 'sugar': 2.29,
    'cereal': 1.79
}
```

Define a function named `customerCartTotal` that takes the name of a customer, a shopping cart dictionary, and a price list dictionary, and returns the total amount for the items in the customer's cart.
The second problem

// to build the dictionary of customers and total costs
function cartTotals(carts, prices) {
    var answer = {};
    for (customer in carts) {  // for each customer in carts
        . . . compute the total for their cart . . .
        . . . add customer : total pair to the answer . . .
    }
    return answer;
}

// to compute the total for a given customer's cart
function customerCartTotal(customer, carts, prices) {
    . . . look up the customer's cart . . .
    . . . the total for that cart . . .
}

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for (var item of cart) {
        var price = prices[item];
        total = total + price;
    }
    return total;
}
Look up a customer's cart

// to compute the total for a given customer's cart
function customerCartTotal(customer, carts, prices) {
    var cart = carts[customer];
    . . . the total for that cart . . .
}

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for (var item of cart) {
        var price = prices[item];
        total = total + price;
    }
    return total;
}
Compute the total for that cart

```javascript
// to compute the total for a given customer's cart
function customerCartTotal(customer, carts, prices) {
  var cart = carts[customer];
  return singleCartTotal(cart, prices);
}

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
  var total = 0;
  for (var item of cart) {
    var price = prices[item];
    total = total + price;
  }
  return total;
}
```
We already did this!

```javascript
// to build the dictionary of customers and total costs
function cartTotals(carts, prices) {
  var answer = {};
  for (var customer in carts) { // for each customer in carts
    . . . compute the total for their cart . . .
    . . . add customer : total pair to the answer . . .
  }
  return answer;
}

// to compute the total for a given customer's cart
function customerCartTotal(customer, carts, prices) {
  var cart = carts[customer];
  return singleCartTotal(cart, prices);
}

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
  var total = 0;
  for (var item of cart) {
    var price = prices[item];
    total = total + price;
  }
  return total;
}
```

😀

We solved this already!
Tackle the original problem again:

// to build the dictionary of customers and total costs

... for each customer in carts ...
... compute the total for their cart ...
... add customer : total pair to the answer ...

// to compute the total for a given customer's cart
function customerCartTotal(customer, carts, prices) {
    var cart = carts[customer];
    return singleCartTotal(cart, prices);
}

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for (var item of cart) {
        var price = prices[item];
        total = total + price;
    }
    return total;
}
exercise 1(c)

Assume you have completed exercise 1(b).

Define a function named `cartTotals` that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for the items in their respective shopping carts.
There are three basic steps

// to build the dictionary of customers and total costs

  ... for each customer in carts ...
  ... compute the total for their cart ...
  ... add customer : total pair to the answer ...

// to compute the total for a given customer's cart
function customerCartTotal(customer, carts, prices) {
    var cart = carts[customer];
    return singleCartTotal(cart, prices);
}

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for (var item of cart) {
        var price = prices[item];
        total = total + price;
    }
    return total;
}
This follows a familiar pattern:

// to build the dictionary of customers and total costs
function cartTotals(carts, prices) {
    var answer = {};
    for (var customer of Object.keys(carts)) {
        . . . compute the total for their cart . . .
        . . . add customer : total pair to the answer . . .
    }
    return answer;
}

// to compute the total for a given customer's cart
function customerCartTotal(customer, carts, prices) {
    var cart = carts[customer];
    return singleCartTotal(cart, prices);
}

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for (var item of cart) {
        var price = prices[item];
        total = total + price;
    }
    return total;
}
Compute the total for a cart...

// to build the dictionary of customers and total costs
function cartTotals(carts, prices) {
    var answer = {};
    for (var customer of Object.keys(carts)) {
        var total = customerCartTotal(customer, carts, prices);
        // . . . add customer: total pair to the answer . . .
    }
    return answer;
}

// to compute the total for a given customer's cart
function customerCartTotal(customer, carts, prices) {
    var cart = carts[customer];
    return singleCartTotal(cart, prices);
}

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for (var item of cart) {
        var price = prices[item];
        total = total + price;
    }
    return total;
}
...which we already know how to do...

```javascript
// to build the dictionary of customers and total costs
function cartTotals(carts, prices) {
    var answer = {};
    for (var customer of Object.keys(carts)) {
        var total = customerCartTotal(customer, carts, prices);
        ... add customer : total pair to the answer ...
    }
    return answer;
}

// to compute the total for a given customer's cart
function customerCartTotal(customer, carts, prices) {
    var cart = carts[customer];
    return singleCartTotal(cart, prices);
}

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for (var item of cart) {
        var price = prices[item];
        total = total + price;
    }
    return total;
}
```

We solved this already too!
...and put the customer and total together in the dictionary!

// to build the dictionary of customers and total costs
function cartTotals(carts, prices) {
    var answer = {};
    for (var customer of Object.keys(carts)) {
        var total = customerCartTotal(customer, carts, prices);
        answer[customer] = total;
    }
    return answer;
}

// to compute the total for a given customer's cart
function customerCartTotal(customer, carts, prices) {
    var cart = carts[customer];
    return singleCartTotal(cart, prices);
}

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for (var item of cart) {
        var price = prices[item];
        total = total + price;
    }
    return total;
}
// to build the dictionary of customers and total costs
function cartTotals(carts, prices) {
    var answer = {};
    for (var customer of Object.keys(carts)) {
        var total = customerCartTotal(customer, carts, prices);
        answer[customer] = total;
    }
    return answer;
}

// to compute the total for a given customer's cart
function customerCartTotal(customer, carts, prices) {
    var cart = carts[customer];
    return singleCartTotal(cart, prices);
}

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for (var item of cart) {
        var price = prices[item];
        total = total + price;
    }
    return total;
}