Road map

▶ Review ◀

dictionaries (key-value mappings)

exercises
A dictionary is a set of key:value pairs, with the requirement that the keys are unique (within one dictionary).
Operations

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

Add/Update
>>> x['c'] = 12
>>> x.update( { 'b':7, 'z':3 } )

Retrieve
>>> x['c']
>>> x.get('c')

Remove
>>> del x['c']
>>> x.pop('z')

Membership
>>> 'c' in x
>>> 'c' not in x
Operations

**Keys**

```python
>>> x.keys()
>>> list(x.keys())
>>> list(x)
>>> sorted(x)
```

**Values**

```python
>>> x.values()
>>> list(x.values())
```

**Pairs**

```python
>>> x.items()
>>> list(x.items())
```
Road map

Review

▶ exercises ◀

dictionaries (key-value mappings)

exercises
exercise 1

Define a function that accepts a dictionary and a value, and returns how many times that value occurs in the dictionary.
Define a function that accepts a dictionary and a value, and returns how many times that value occurs in the dictionary.

def valueCount(dict, val):
    answer = 0
    for v in dict.values():
        if v==val:
            answer = answer + 1
    return answer
Define a function that accepts a dictionary and a value, and returns how many times that value occurs in the dictionary.

```python
def valueCount(dict, val):
    answer = 0
    for v in dict.values():
        if v == val:
            answer = answer + 1
    return answer
```

exercise 1

This problem fits the same general pattern we've seen previously!
Define a function that accepts a dictionary and a value, and returns how many times that value occurs in the dictionary.

```python
def valueCount(dict, val):
    answer = 0
    for v in dict.values():
        if v == val:
            answer = answer + 1
    return answer
```

**exercise 1**

*The twist?*

We only conditionally add to `answer`!
exercise 2

Define a function that accepts a dictionary and a value, and returns a list of the keys that are paired with that value in the dictionary.
exercise 2

Define a function that accepts a dictionary and a value, and returns a list of the keys paired with that value in the dictionary.

def matchedKeys(dict, val):
    answer = []
    for k in dict.keys():
        if dict.get(k) == val:
            answer.append(k)
    return answer

This problem also fits the same general pattern we've seen previously!
exercise 2
using "deconstructive assignment":

Define a function that accepts a dictionary and a value, and returns a list of the keys paired with that value in the dictionary.

```python
def matchedKeys(dict, val):
    answer = []
    for it in dict.items():
        k, v = it
        if v == val:
            answer.append(k)
    return answer
```

This problem also fits the same general pattern we've seen previously!
Road map

Review

dictionaries (key-value mappings)

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

Add/Update
> x['c'] = 12
> x['b'] = 7
> x['z'] = 3
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
> x.c
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
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.
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;
}
```
Road map

Review

exercises

dictionaries (key-value mappings)

▶ exercises ◀
exercise 1

Define a function that accepts a dictionary and a value, and returns how many times that value occurs in the dictionary.
exercise 2

Define a function that accepts a dictionary and a value, and returns an array of the keys that are paired with that value in the dictionary.