CSE 115
Introduction to Computer Science I
See pinned post in Piazza to register your interest

FRESHMAN TOWNHALL
Tuesday October 2
5:00 – 7:00 PM
O’Brian 112
with CSE Chair Dr. Chunming Qiao
Ask questions
Learn about CSE
Give feedback
Free pizza
Road map

▶️ Review ◀️

dictionaries (key-value mappings)

live coding

exercises
Patterns in code

function everyOther(s) {
  var result = "";
  for (var i = 0; i < s.length; i = i + 2){
    result = result + s[i];
  }
  return result;
}

function sumTo(max) {
  var sum = 0;
  for (var i=1; i<=max; i=i+1){
    sum = sum + i;
  }
  return sum;
}

function factorial(n) {
  var result = 1;
  for (var i = 1; i <= n ; i = i + 1){
    result = result * i;
  }
  return result;
}

An accumulator variable is initialized to the identity element of the accumulation operator of... used in the loop to construct the answer incrementally.
The final answer is returned.
exercise 1

Write a function named 'implode' that takes an array (JS) or list (Python) of strings and returns a single string consisting of the individual characters of all of the argument strings, in order.

For example, `implode(['a', 'b', 'c'])` must return 'abc'.

Come up with additional test cases.

What is the accumulation operation?

What is the identity element of that operation?
test cases:
implode(['a','b','c']) => 'abc'
implode(['a']) => 'a'
implode([]) => ''

Operation: + (string concatenation)
Initial result value: "" (empty string)

```javascript
function implode(x) {
  var result = "";
  for (var v of x) {
    result = result + v;
  }
  return result;
}
```

```python
def implode(x):
    result ="
    for v in x:
        result = result + v
    return result
```
Write a function named 'explode' that takes a string and returns an array (JS) or list (Python) consisting of the individual characters of the argument string.

For example, explode("abc") must return ["a", "b", "c"].

Come up with additional test cases.

What is the accumulation operation?

What is the initial value for the accumulation variable?

[CURVEBALL: in this case it's not the identity of the operation]
Possible solutions

test cases:
explode('abc') => ['a','b','c']
explode('a') => ['a']
explode('') => []

explode(implode(['a','b','c'])) => ['a','b','c']
implode(explode('abc')) => 'abc'

Operation: push (JS) or append (Python)
Initial result value: [] (empty array/list)

**JS**

```javascript
function explode(str) {
  var x = [];
  for (var v of str) {
    x.push(v);
  }
  return x;
}
```

**Python**

```python
def explode(str):
  x = []
  for v in str:
    x.append(v)
  return x
```
fun application

Open the JavaScript console in your browser.

```javascript
for (var elem of document.all) {
    elem.style.fontFamily = "courier";
}
```
Road map

Review

▶ dictionaries (key-value mappings) ◀

live coding

exercises
key-value mappings

name $\rightarrow$ phone number

UBIT $\rightarrow$ name

person number $\rightarrow$ student record

driver license $\rightarrow$ driving record

etc.
key-value mappings

In Python a key-value mapping is called a dictionary:

*Dictionary are sometimes found in other languages as “associative memories” or “associative arrays”.*
key-value mappings

... dictionaries are indexed by keys ...

strings ... can always be keys.

A dictionary [is] a set of key:value pairs, with the requirement that the keys are unique (within one dictionary).
key-value mappings

A pair of braces creates an empty dictionary: `{}`.

```python
>>> a = {}
>>> a
{}
```

Placing a comma-separated list of key:value pairs within the braces adds initial key:value pairs to the dictionary; this is also the way dictionaries are written on output.

```python
x = {'a': 1, 'b': 2, 'c': 3, 'd': 4}
>>> x
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
```
key-value mappings

Use bracket notation to add/update an entry in an existing dictionary:

\[
x['key'] = value
\]

```python
>>> x['c'] = 12
>>> x
{'a': 1, 'b': 2, 'c': 12, 'd': 4}
```

Can use update function on an existing dictionary to add/update from an existing dictionary:

\[
x.update(y)
\]

```python
x.update( { 'b':7, 'z':3 } )
```

```python
>>> x.update( { 'b':7, 'z':3 } )
>>> x
{'a': 1, 'b': 7, 'c': 12, 'd': 4, 'z': 3}
```
key-value mappings

Access: x[key], x.get(key) —> diff if key not in x

```python
>>> x['c']
12
>>> x.get('c')
12
>>> x['q']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'q'
>>> x.get('q')
>>> x.get('q',False)
False
```
key-value mappings

Remove: `del x[key], x.pop(key)`

```python
>>> del x['c']
>>> x
{'a': 1, 'b': 7, 'd': 4, 'z': 3}
>>> x.pop('z')
3
>>> x
{'a': 1, 'b': 7, 'd': 4}
```
key-value mappings

Sequences: \(x\text{.keys()}\), \(x\text{.values()}\), \(x\text{.items()}\)

Lists: \(\text{list}(x)\), \(\text{sorted}(x)\), or \(\text{list}(x\text{.keys}())\), \(\text{list}(x\text{.values}())\), \(\text{list}(x\text{.items}())\)

Membership test: key in \(x\), key not in \(x\)
Road map

Review
dictionaries (key-value mappings)

▶ live coding ◀
exercises
Alphonce's live coding transcript

user@4444c158eae7:/projects$ python3
Python 3.5.1 (default, Jul 5 2018, 13:06:10)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> colors = {}
>>> colors
{}
>>> colors['Jonathan'] = 'black'
>>> colors
{'Jonathan': 'black'}
>>> colors['Cassandra'] = 'blue'
>>> colors
{'Cassandra': 'blue', 'Jonathan': 'black'}
>>> colors.update({'Snigdha': 'turquoise', 'Alex': 'green', 'Ann': 'yellow'})
>>> colors
{'Alex': 'green', 'Snigdha': 'turquoise', 'Cassandra': 'blue', 'Jonathan': 'black', 'Ann': 'yellow'}
>>> 'Carl' in colors
False
>>> 'Snigdha' in colors
True
>>> 'turquoise' in colors
False
>>> colors.get('Snigdha')
'turquoise'
>>> colors.get('Carl')
None
>>> colors.get('turquoise')
>>> colors['Carl']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    KeyError: 'Carl'
>>> colors.keys()
dict_keys(['Alex', 'Snigdha', 'Cassandra', 'Jonathan', 'Ann'])
>>> list(colors.keys())
['Alex', 'Snigdha', 'Cassandra', 'Jonathan', 'Ann']
>>> colors.values()
dict_values(['green', 'turquoise', 'blue', 'black', 'yellow'])
>>> list(colors.values())
['green', 'turquoise', 'blue', 'black', 'yellow']
>>> colors.items()
dict_items([('Alex', 'green'), ('Snigdha', 'turquoise'), ('Cassandra', 'blue'), ('Jonathan', 'black'), ('Ann', 'yellow')])
>>> colors['Carl'] = 'it depends'
>>> colors
{'Cassandra': 'blue', 'Carl': 'it depends', 'Alex': 'green', 'Snigdha': 'turquoise', 'Jonathan': 'black', 'Ann': 'yellow'}
>>> colors['Carl'] = 'blue'
>>> colors
{'Cassandra': 'blue', 'Carl': 'blue', 'Alex': 'green', 'Snigdha': 'turquoise', 'Jonathan': 'black', 'Ann': 'yellow'}
>>> colors.values()
dict_values(['blue', 'blue', 'green', 'turquoise', 'black', 'yellow'])
>>>

...
Road map

Review
dictionaries (key-value mappings)

live coding

▶ 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 a list of the keys paired with that value in the dictionary.