CSE 115
Introduction to Computer Science I
Road map

▶ Review ◀

File reading

newline

exercise


**exercise: complex problem**

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.
Decompose into subproblems...

// to build the dictionary of customers and total costs
function cartTotals(carts, prices) {
    var answer = {};
    for ( ... 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 ...
    return ... compute the total for that cart ...
}

// to compute the total for a single cart
function singleCartTotal(cart, prices) {
    var total = 0;
    for ( ... for each item in cart ... ) {
        ... look up the cost of that item in prices ...
        ... add the cost to total ...
    }
    return total;
}
...and compose solution.

```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);
        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;
}
```
Road map

Review

▶️ File reading ◀️

newline

exercise
File reading

A bit of text

on several lines

A text file is a sequence of characters.

The contents can be read line by line:

A bit of text

on several lines

...
7.2. Reading and Writing Files

open() returns a file object, and is most commonly used with two arguments: open(filename, mode).

>>> f = open('workfile', 'w')
7.2. Reading and Writing Files

`open()` returns a file object, and is most commonly used with two arguments: `open(filename, mode)`.

```python
>>> f = open('workfile', 'w')
```

The first argument is a string containing the filename. The second argument is another string containing a few characters describing the way in which the file will be used. `mode` can be `'r'` when the file will only be read, `'w'` for only writing (an existing file with the same name will be erased), and `'a'` opens the file for appending; any data written to the file is automatically added to the end. [...] The mode argument is optional; `'r'` will be assumed if it’s omitted.
7.2. Reading and Writing Files

`open()` returns a file object, and is most commonly used with two arguments: `open(filename, mode)`.

```python
>>> f = open('workfile', 'w')
```

The first argument is a string containing the filename. The second argument is another string containing a few characters describing the way in which the file will be used. `mode` can be `r` when the file will only be read, `w` for only writing (an existing file with the same name will be erased), and `a` opens the file for appending; any data written to the file is automatically added to the end. [...] The mode argument is optional; `'r'` will be assumed if it’s omitted.

Normally, files are opened in text mode, that means, you read and write strings from and to the file, which are encoded in a specific encoding. [...]

In text mode, the default when reading is to convert platform-specific line endings (`\n` on Unix, `\r\n` on Windows) to just `\n`. When writing in text mode, the default is to convert occurrences of `\n` back to platform-specific line endings. [...]

7.2. Reading and Writing Files

`open()` returns a file object, and is most commonly used with two arguments: `open(filename, mode)`.

```python
>>> f = open('workfile', 'w')
```

The first argument is a string containing the filename. The second argument is another string containing a few characters describing the way in which the file will be used. `mode` can be `'r'` when the file will only be read, `'w'` for only writing (an existing file with the same name will be erased), and `'a'` opens the file for appending; any data written to the file is automatically added to the end. [...] The mode argument is optional; `'r'` will be assumed if it’s omitted.

Normally, files are opened in text mode, that means, you read and write strings from and to the file, which are encoded in a specific encoding. [...] In text mode, the default when reading is to convert platform-specific line endings (\n on Unix, \r\n on Windows) to just \n. When writing in text mode, the default is to convert occurrences of \n back to platform-specific line endings. [...] It is good practice to use the `with` keyword when dealing with file objects. The advantage is that the file is properly closed after its suite finishes, even if an exception is raised at some point. [...]
File reading

File reading is easily handled using a 'with...as' statement:

```python
with open("Chapter1.txt") as f:
    ...do something with file...
```

f is a variable. It refers to a file object.
File reading

File objects support iteration:

```python
with open("Chapter1.txt") as f:
    for line in f:
        ...do something with each line...
```
File reading

File objects support iteration:

```python
with open("Chapter1.txt") as f:
    for line in f:
        print(line)
```
Printing

File objects support iteration:

```python
with open("Chapter1.txt") as f:
    for line in f:
        print(line)
```
Lines read have newline

A bit of text

on several lines

A text file is a sequence of characters.

The contents can be read line by line:
File objects support iteration:

```python
with open("Chapter1.txt") as f:
    for line in f:
        print(line)
```

Either remove newline from line, or don't print extra newline when printing.
Printing without extra newline

File objects support iteration:

```python
with open("Chapter1.txt") as f:
    for line in f:
        print(line, end=""")
```

Try different end strings.
Removing extra newline

File objects support iteration:

```python
with open("Chapter1.txt") as f:
    for line in f:
        line = line.rstrip(\"\r\n\")
        print(line)
```

"Removes" any end-of-line characters, in any order, from right edge of string.
Removing extra newline

File objects support iteration:

```python
with open("Chapter1.txt") as f:
    for line in f:
        line = line.rstrip(\r\n)
        print(line)
```

A new string is created without any end-of-line characters, in any order, at right edge of string.
with open("Chapter1.txt") as f:
    count = 0
    for line in f:
        line = line.rstrip('\r\n')
        count = count + 1
        print('Line #{0}: {1}'.format(count, line))

{0} and {1} are placeholders
with open("Chapter1.txt") as f:
    count = 0
    for line in f:
        line = line.rstrip("\r\n")
        count = count + 1
        print('Line #{0:03d}: {1}'.format(count, line))

For more information:
https://docs.python.org/3/library/stdtypes.html#str.format
Exercises

1. Define a function that takes a file name as an argument and returns a map with character counts for the file.

2. Define a function that takes a file name as an argument and returns a map with word counts for the file.

Q: What counts as a word?

Q: How do we segment a string into words?