Dictionaries

CMPT 140

Dictionaries

{ key : value }

Exercise 1

Suppose dictionary `schedule` represents an employee's work schedule where the **keys** are a day of the week and the **values** are the number of hours scheduled that day. Write Processing code to:

(a) Create `schedule` such that the employee works 8 hours on Monday, 7.5 hours on Tuesday, 8 hours on Thursday, and 7.5 hours on Friday.

Exercise 1

Suppose dictionary `schedule` represents an employee's work schedule where the keys are a day of the week and the values are the number of hours scheduled that day. Write Processing code to:

(a) Create `schedule` such that the employee works 8 hours on Monday, 7.5 hours on Tuesday, 8 hours on Thursday, and 7.5 hours on Friday.

(b) Oops! Someone made a scheduling error. Update `schedule` so that the employee works Friday's hours on Wednesday instead. Remove Friday from the work schedule.

Exercise 2

We have a dictionary called `poll`, representing employee feedback data:

- **keys:** employee usernames
- **values:** one of three possible string values: "Computers", "Seating" or "Support"

Write function `poll_results` which takes dictionary parameter `poll` and **returns** a new dictionary whose keys are "Computers", "Seating", and "Support" and values are the number of employees who requested that improvement. Only create the keys in the new dictionary if at least one employee requested it!

Exercise 2

Example `poll` input:

```
poll = {  "ab"   :  "Computers",
          "cc"   :  "Support",
          "bar"  :  "Computers" }
```

Expected output:

```
{  "Computers"  :  2,
   "Support"    :  1 }
```

Exercise 3

Is the following dictionary an example of a mapping or record? Explain why.

```
song = {  "artist"  :  "Elvis Presley",
          "title"   :  "Jailhouse Rock",
          "length"  :  146 }
```

Ciphers

In cryptography, messages are encoded and decoded using ciphers to transfer private information between two parties. Ciphers are legends that denote what character should be substituted by another in an encoded message in order to obtain the original (decoded) message.

Example encoded message and a **decoding** cipher:

```
msg = "SVJJT"
cipher = { "J":"L", "S":"H", "T":"O", "V":"E" }
```

Message decoded using cipher:

```
decoded_msg = "HELLO"
```

Exercise 4

Write Processing function `invert_cipher` which takes an **encoding cipher** (i.e. that maps original characters to their encoded character) and returns a **decoding cipher** (i.e. that maps encoded characters to original characters).

Example cipher input:

```
{ "A":"Z", "B":"W", ... }
```

Expected return data:

```
{ "Z":"A", "W":"B", ... }
```

Exercise 5

Suppose we receive a secret message that we want to decode. Write Processing function `decode` which takes string `encoded_msg` and a **decoding** `cipher` and returns the decoded message. Characters unknown to the cipher remain the same in both messages.

Example encoded message and cipher input:

```
encoded_msg = "LOLOL!"
cipher = { "L":"H", "O":"A", ... }
```

Expected decoded message as output:

```
"HAHAH!"
```

Exercise 6

Let's redo the planet simulation from Chapter 14, except:

- we'll store planet information using a **list of records** instead of multiple lists
- every planet will have a different size
- new planets from a mouse click will have a random size