20. Dictionaries

Topics:
- Basic dictionary manipulations
- How they are different from lists
- Application: Word frequency in the Sonnet Collection
A First Example


This dictionary has 5 items:

‘I’ : 1
‘V’ : 5
‘X’ : 10
‘L’ : 50
‘C’ : 100
An item has a **key** and a **value**.

For the item `‘V’:5`,

- `'V'` is the key
- `5` is the value
Set-Up

\[ D = \{ 'I':1, 'V':5, 'X':10, 'L':50, 'C':100 \} \]

To set up a small dictionary in this style you:

1. Use a colon to separate a key from its value.
2. Separate items with a comma.
3. Enclose the whole thing with curly brackets.
Some Questions

How do you see if a dictionary has a key?
How do you access items in a dictionary?
How can you add an item to a dictionary?
How is a dictionary different from a list?
Are there type-related rules about keys?
Are there type-related rules about values?
Checking to see if a Dictionary Has a Particular Key

```python
>>> D = {'I':1,'V':5,'X':10}
>>> 'I' in D
True
>>> 'II' in D
False
```
Checking if D has a particular Value

Produce a list of all the values in D.

Then use “in” on that list

```python
>>> D = {'I':1,'V':5,'X':10}
>>> L = D.values()
>>> L
[1, 10, 5]
>>> 5 in L
True
```
Extracting a Value

>>> D = {'I':1,'V': 5,'X':10}
>>> a = D['V']
>>> a
5

Use square bracket notation.

Use the key not an integer subscript.
Adding an Item to a Dictionary

```python
>>> D = {'I':1,'V':5,'X':10}
>>> D['C'] = 100
>>> D
{'I': 1, 'X': 10, 'C': 100, 'V': 5}
```
Cannot Have Multiple Keys

This modifies an existing item:

```python
>>> D = {'I':1,'V':5,'X':10}
>>> D['I'] = 100
>>> D
{'I': 100, 'X': 10, 'V': 5}
```

We do not produce:

```
D = {'I':1,'V':5,'X':10, 'I':100}
```
Dictionaries are Different From Lists

```python
>>> D = {'I':1, 'V':5, 'X':10, 'L':50}
>>> D
{'I': 1, 'X': 10, 'L': 50, 'V': 5}
```

The items in a dictionary are not ordered as in a list.

We see here that Python "shows" a different ordering than how D was set up.
Dictionaries are Different From Lists

Dictionary values are accessed by key not subscript.

```python
>>> D = {'I': 1, 'X': 10, 'V': 5}
```

```python
>>> D['X']
10
```

```python
>>> L = [1,5,10]
```

```python
>>> L[1]
5
```
Dictionaries are Different From Lists

Dictionary values are accessed by key not subscript.

```python
>>> D = {'I': 1, 'V': 5, 'X': 10}
>>> D[2]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 2
```

Python is complaining because 2 is not a key in the D
Lists and Dictionaries

**Lists involve mappings from ints to values**

**Dictionaries involve mappings from keys to values**

```
>>> x = []
>>> x.append(3)
>>> x.append(5)
>>> x.append(1)
```

```
>>> D = {}
>>> D['I'] = 1
>>> D['V'] = 5
>>> D['X'] = 10
```
Lists and Dictionaries

You “add” to a list using the `append` method.
You add an item to a dictionary using a “new” key.
Lists and Dictionaries

L = [ ] and L = list() are equivalent
D = { } and D = dict() are equivalent

>>> L = []
>>> L.append(3)
>>> L.append(5)
>>> L.append(1)

>>> D = {}
>>> D[ 'I' ] = 1
>>> D[ 'V' ] = 5
>>> D[ 'X' ] = 10

Empty List
Empty Dict
Dictionaries & Lists

Square Bracket Notation

\[ D[ 'x' ] \quad L[2] \]

The \texttt{len} function

\[ \text{len}(D) \quad \text{len}(L) \]

So, of course, there are some similarities between lists and dictionaries.
For-Loops and Dictionaries

\[
D = \{ 'I': 1, 'V': 5, 'X': 10, 'L': 50 \}
\]

for d in D:
    print d, D[d]

Again, dictionaries are not ordered. So extra steps would need to be taken here for things to be printed in a certain order.
Pretty Printing a Short Dictionary

```python
>>> D = {'I':1,'V':5,'X':10,'L':50}
>>> str(D)
"{'I': 1, 'X': 10, 'L': 50, 'V': 5}"```
Other Examples and Rules

D1 = {"red":[1,0,0],"cyan":[0,1,1]}

D2 = {1:’one’, 2:’two’, 3:’three’}

D3 = {’A’:Point(1,2),’B’:Point(3,4)}

D4 = {’A’:'B', 1:'C', ‘D’:2}

- Keys must be strings or numbers
- Values can be anything
- Typically the items all “look alike”, but not nec.
Some Common Errors

```python
>>> D = {'I':1,'V':5,'X':10}
>>> D('I')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'dict' object is not callable
```

Square brackets, not parens!
Some Common Errors

```python
>>> D = {'I': 1, 'X': 10, 'V': 5}
```

```python
>>> x = D['L']
```

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'L'

Trying to access a nonexistent item.

Note: `D['L'] = 50` is legal and adds an item to `D`
A More Involved Dictionary Problem

How many times do each of the following words occur in the Shakespeare Sonnet Collection?

love    sun     moon   sad
happy   thou    me     rain
flowers water   dude
Clouds   wonder  forever
Overall Plan

Use a dictionary D of counters

The keys will be words

The values will be ints that keep track of frequency.
We go through the sonnets word-by-word.

If a word $w$ is already a key, increment the corresponding value, i.e.,

$$D[w] += 1$$

If the word $w$ is not a key, then add it to $D$ and initialize its corresponding value, i.e.,

$$D[w] = 1$$
D = { 'sun':34, 'moon':5 , 'thou':56 }

This would “say” that there are

34 occurrences of ‘sun’,
5 occurrences of ‘moon’, and
56 occurrences of ‘thou’.
Updating a Dictionary

\[ W = [\text{'cat'}, \text{mouse'}, \text{'dog'}, \text{'cat'}, \text{rabbit'}] \]

D ---\(\rightarrow\) 'cat' \(\rightarrow\) 20
D ---\(\rightarrow\) 'dog' \(\rightarrow\) 10

Look at each word in W and update D accordingly.
Updating a Dictionary

\[ W = \{\text{'cat'}, \text{mouse'}, \text{'dog'}, \text{'cat'}, \text{rabbit'}\} \]

Before

D \longrightarrow \begin{align*}
\text{'cat'} & \longrightarrow 20 \\
\text{'dog'} & \longrightarrow 10
\end{align*}

Look at each word in W and update D accordingly
Updating a Dictionary

\[ W = ['\text{cat}', \text{mouse}', '\text{dog}', '\text{cat}', \text{rabbit'}] \]

Look at each word in W and update D accordingly.
Updating a Dictionary

$W = \['\text{cat}', '\text{mouse}', '\text{dog}', '\text{cat}', '\text{rabbit}'\]$
Updating a Dictionary

\[ W = [\text{'cat'}, \text{'mouse'}, \text{'dog'}, \text{'cat'}, \text{'rabbit'}] \]

\[ D \rightarrow \]

\begin{align*}
\text{'cat'} & \rightarrow 21 \\
\text{'dog'} & \rightarrow 10 \\
\text{'mouse'} & \rightarrow 1 \\
\end{align*}

After

Look at each word in \( W \) and update \( D \) accordingly.
Updating a Dictionary

$W = ['\text{cat}', '\text{mouse}', '\text{dog}', '\text{cat}', '\text{rabbit}']$

Look at each word in $W$ and update $D$ accordingly.

Before

\[\begin{array}{c|c}
\text{cat} & 21 \\
\text{dog} & 10 \\
\text{mouse} & 1 \\
\end{array}\]
Updating a Dictionary

$W = ['cat', 'mouse', 'dog', 'cat', 'rabbit']$

Look at each word in $W$ and update $D$ accordingly.

After

$D ----> 'cat' --> 21$
$D ----> 'dog' --> 11$
$D ----> 'mouse' --> 1$
Updating a Dictionary

\[ W = ['\text{cat}', '\text{mouse}', '\text{dog}', '\text{cat}', '\text{rabbit}'] \]

\[ D \rightarrow \begin{align*}
\text{'cat'} & \rightarrow 21 \\
\text{'dog'} & \rightarrow 11 \\
\text{'mouse'} & \rightarrow 1
\end{align*} \]

Before

Look at each word in \( W \) and update \( D \) accordingly.
Updating a Dictionary

\[ W = ['\text{cat}', 'mouse', 'dog', 'cat', 'rabbit'] \]

\[ D \rightarrow \]

\[ 'cat' \rightarrow 22 \]
\[ 'dog' \rightarrow 11 \]
\[ 'mouse' \rightarrow 1 \]

After

Look at each word in \( W \) and update \( D \) accordingly.
Updating a Dictionary

W = ['cat', 'mouse', 'dog', 'cat', 'rabbit']

D -->

'cat' --> 22
'dog' --> 11
'mouse' --> 1

Before

Look at each word in W and update D accordingly.
Updating a Dictionary

W = ['cat', 'mouse', 'dog', 'cat', 'rabbit']

Look at each word in W and update D accordingly.
From the A6 Module
SonnetTools.py we use

GetSonnets()
   Reads all the sonnets from a text file and stores each line in a list of strings

dePunc(s)
   Removes all punctuation from string s
The Function GetSonnets()

Returns a list of strings.

Each string is a sonnet line, or a blank line, or an index line.

```python
>>> L = GetSonnets()
>>> len(L)
2584
>>> L[289]
'XVIII.'
>>> L[291]
"Shall I compare thee to a summer's day?"
```
The Function dePunc

Removes all punctuation...

```python
>>> s = 'a.b,c?d!f:g;'
>>> t = dePunc(s)
>>> t
'abcdfg'
```
We Write Three Functions

**WordsInLine(s)**

Takes a sonnet line and returns a list of its words.

**UpdateFreqD(D,w)**

Either adds word w to the dictionary of counters D or increments D[w].

**MakeFreqD(L)**

Returns a dictionary of counters based on All the sonnets encoded in the list L.
Getting the Words in a String

```python
def WordsInLine(s):
    s = s.lower()
    s = dePunc(s)
    W = s.split()
    return W

>>> a = 'One, Two, Three. GO!'
>>> WordsInLine(a)
['one', 'two', 'three', 'go']
```

Returns a list of all the words in s
The split Method

```python
>>> a = 'One Two Three GO'
>>> b = a.split()
>>> b
['One', 'Two', 'Three', 'GO']
```
Updating a Dictionary of Counters

```python
def UpdateFreqD(D,s):
    if s in D:
        D[s] +=1
    else:
        D[s] = 1

>>> D = {'x':10,'y':20,'z':30}

>>> UpdateFreqD(D,'y')

>>> D
{'y': 21, 'x': 10, 'z': 30}
```
Updating D

def UpdateFreqD(D,s):
    if s in D:
        D[s] += 1
    else:
        D[s] = 0

>>> D = {'x':10,'y':20,'z':30}
>>> UpdateFreqD(D,'w')
>>> D
{'y': 20, 'x': 10, 'z': 30, 'w': 0}
Making a Frequency Dictionary

def MakeFreqD(L):
    """ L is a list of sonnet line strings """
    D = dict()
    for s in L:
        W = WordsInLine(s)
        # W is a list of the words
        # in line s
        for w in W:
            UpdateFreqD(D,w)
    return D
Some Frequencies

love 162
sun 11
moon 3
sad 7
happy 11
thou 229
me 164
flowers 7
water 5
dude 0
rain 3
clouds 4
wonder 3
forever 0