16. Dictionaries

Topics:
- Basic dictionary manipulations
- How they are different from lists
- Dictionaries are Objects
- Application: Word frequency

A First Example

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

This dictionary has 5 items:

```
'I':1
'V':5
'X':10
'L':50
'C':100
```

Keys and Values

```
D = {'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 do this:

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.

The Methods `.keys` and `.values`

```
>>> D =
{'I':1,'V':5,'X':10,'L':50,'C':100}

>>> D.keys()
['I', 'X', 'C', 'L', 'V']

>>> D.values()
[1, 10, 100, 50, 5]
```

Creates a list of all the keys

Creates a list of all the values

Deleting a Dictionary Item with `del`

```
>>> D = {'I':1,'V':5,'X':10,'L':50,'C':100}

>>> del D['X']

>>> D
{'I':1,'V':5,'L':50,'C':100}
```

Some Questions

How do you check 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

```
>>> D = {'I':1,'V':5,'X':10}
>>> 'I' in D
True
>>> 'II' in D
False
>>>
```

Use `in` to check if a dictionary has a particular key.

Checking if D has a particular Value Using the `values` Method

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

Produce a list of all the values in D

Use "in" on that list

Extracting a Value

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

Use square bracket notation.

Use the key as you would an integer subscript.

Adding an Item to a Dictionary

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

Use assignment, e.g., `D['C'] = 100`

This "connects" the assigned value to the key named within square brackets making the pair an item, e.g., `'C': 100`

Changing the Value of an Item

```
>>> D = {'I':1,'V':5,'X':10}
>>> D['V'] = 55
>>> D
{'I':1, 'X':10, 'V':55}
```

`D['V'] = 55` does not produce

`{'I':1, 'V':5, 'X':10, 'V':55}`

Cannot have two items with the same key.

Dictionaries are Different from Lists

```
>>> 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.

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

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

Dictionary

List

Dictionaries are Different From Lists

Dictionary values are accessed by key not subscript.

```
>>> 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

```
x --->  0 --->  3
        1 --->  5
        2 --->  1
```

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

```
D --->  'I' --->  1
        'V' --->  5
        'X' --->  10
```

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

Lists involve mappings from ints to values
Dictionaries involve mappings from keys to values

Lists and Dictionaries

```
x --->  0 --->  3
        1 --->  5
        2 --->  1
```

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

```
D --->  'I' --->  1
        'V' --->  5
        'X' --->  10
```

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

You "add" to a list using the append method.
You add an item to a dictionary using a "new" key.

Lists and Dictionaries

```
x --->  0 --->  3
        1 --->  5
        2 --->  1
```

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

Empty List

```
D --->  'I' --->  1
        'V' --->  5
        'X' --->  10
```

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

Empty Dict

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

Dictionaries & Lists

Access via the Square Bracket Notation:

```
D['x']          L[2]
```

The `len` function can be applied to both:

```
>>> x = [10,20,30]
>>> len(x)
3
>>> D = {'a':10,'b':20,'c':30}
>>> len(D)
3
```

Dictionaries & Lists Are Objects

```
>>> x = [10,20,30]
>>> y = x
>>> x[0]=100
>>> y
[100, 20, 30]
```

You can have multiple references to the same object. This is the idea of an alias.

```
>>> D = {'a':10,'b':20,'c':30}
>>> E = D
>>> D['a'] = 100
>>> E
{'a': 100, 'c': 30, 'b': 20}
```

Dictionaries & Lists Are Objects

```
>>> x = [10,20,30]
>>> y = list(x)
>>> x[0] = 100
>>> y
[10, 20, 30]
```

It is possible to make copies.

```
>>> D = {'a':10,'b':20,'c':30}
>>> E = dict(D)
>>> D['a']=100
>>> E
{'a': 10, 'c': 30, 'b': 20}
```

For-Loops and Dictionaries

```
D = {'I':1,'V':5,'X':10,'L':50}
for d in D:
   print d, D[d]
```

```
I    1
X   10
L   50
V    5
```

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

For-Loops and Dictionaries

```
D = {'I':1,'V':5,'X':10,'L':50}
KeysOfD = D.keys()
KeysOfD.sort()
for d in KeysOfD:
   print d, D[d]
```

```
I    1
L   50
V    5
X   10
```

Pretty Printing a Short Dictionary

```
>>> 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':'B', 1:'C', 'D':2}
```

- Keys must be strings or numbers
- Values can be anything
- Typically the items all "look alike", but that is not necessary.

Some Common Errors

```
>>> 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

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

>>> 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 Dictionary Application

Given a text file **F** that encodes (for example) an English novel, create a dictionary **D** that specifies the frequency of each word that appears in the file.

Word Frequency Dictionaries

The dictionary

```
D = {'sun':34,'moon':5,'darcy':56}
```

would "say" that there are

34 occurrences of 'sun',
5 occurrences of 'moon', and
56 occurrences of 'darcy'.

Strategy

First, read the file and create a list of strings, one string for each line in the file:

```
L = FileToList('PridePrej.txt')
```

Strategy

Second, assume the existence of a function that can extract a list of words a string s and use it like this:

```
wList = stringToWordList(s)
```

Thus, we would get

```
['the','quick','brown','fox']
```

from

```
'The quick, brown fox!'
```

Strategy

Third, figure out how to update the word frequency dictionary D:

```
L = fileToStringList('PridePrej.txt')
D = {}
for s in L:
    wList = stringToWordList(s)
    for w in wList:
        Update(w,D)
```

Updating a Dictionary

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

```
D ---> 'cat'    --->  20
       'dog'    --->  10
```

Look at each word in W and update D accordingly

Updating a Dictionary

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

```
D ---> 'cat'    --->  20
       'dog'    --->  10
```

Before

Look at each word in W and update D accordingly

Updating a Dictionary

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

```
D ---> 'cat'    --->  21
       'dog'    --->  10
```

After

Look at each word in W and update D accordingly

Updating a Dictionary

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

```
D ---> 'cat'    --->  21
       'dog'    --->  10
```

Before

Look at each word in W and update D accordingly

Updating a Dictionary

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

D ---> 'cat'    ---> 21
       'dog'    ---> 10
       'mouse'  --->  1
```

After

Look at each word in W and update D accordingly

Updating a Dictionary

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

D ---> 'cat'    ---> 21
       'dog'    ---> 10
       'mouse'  --->  1
```

Before

Look at each word in W and update D accordingly

Updating a Dictionary

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

D ---> 'cat'    ---> 21
       'dog'    ---> 11
       'mouse'  --->  1
```

After

Look at each word in W and update D accordingly

Updating a Dictionary

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

D ---> 'cat'    ---> 21
       'dog'    ---> 11
       'mouse'  --->  1
```

Before

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
```

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']
```

```
D --->  'cat'     --->  22
        'dog'     --->  11
        'mouse'   --->   1
        'rabbit'  --->   1
```

After

Look at each word in W and update D accordingly

We Design Two Key Functions To Do all the Work

```
stringToWordList(s)

Update(wList,D)
```

stringToWordList

```
def stringToWordList(s):
    t = ''
    for c in s.lower():
      alfa = 'abcdefghijklmnopqrstuvwxyz'
        if c in alfa:
            t = t + c
        else:
            t = t + ' '
    return t.split()
```

A word is made up of lower case letters.

After the loop, words are separated by blanks in the string t.

Update(wList,D)

```
def Update(wList,D):
    for w in wList:
        z = w.lower()
        if z in D:
            D[z]+=1
        else:
            D[z]=1
```

z is in the dictionary. So increase its frequency count

z is not in the dictionary. So add it in with frequency initialized to 1

A Sample Computation

```
communicativeness
condescendingly
conscientiously
disappointments
discontentedness
disinterestedness
merchantibility
misrepresentation
recommendations
representations
superciliousness
superintendence
uncompanionable
unenforceability
```

These are all the words in Pride and Prejudice that occur only once and which have 15 or more Letters.

Method. Compute the complete word frequency dictionary. Then go through it printing a key if its value is 1 and its length is 15 or greater.