Dictionary

In [ ]:
```
"""
Write a function wordcount() that:

(1) Takes as input a text—as a string; and
(2) Prints the frequency of each word in the text.
Assume there is no punctuation in the text.

>>> text = 'all animals are equal but some animals are more equal than other'

>>> wordCount(text)
all appears 1 time.
animals appears 2 times.
some appears 1 time.
equal appears 2 times.
but appears 1 time.
other appears 1 time.
are appears 2 times.
than appears 1 time.
more appears 1 time.
"""
```

In [ ]:
```
def wordcount(text):
    dict = {}
    wordlst = text.split()
    for w in wordlst:
        if w not in dict:
            dict[w] = 1
        else:
            dict[w] += 1
    for k in dict.keys():
        if dict[k] == 1:
            print('{} appears {} time.'.format(k, dict[k]))
        else:
            print('{} appears {} times.'.format(k, dict[k]))
```

In [2]:
```
text = 'all animals are equal but some animals are more equal than other'
```

In [3]:
```
wordcount(text)
```

```
all appears 1 time.
animals appears 2 times.
are appears 2 times.
equal appears 2 times.
but appears 1 time.
some appears 1 time.
more appears 1 time.
than appears 1 time.
other appears 1 time.
```

In [4]:
```
dict = {'a':1, 'b':2, 'c':3, 'd':4}
```

In [5]:
```
for k in dict:
    print(k)
```

```
a
b
c
d
```

In [6]:
```
for k in dict.keys():
    print(k)
```

```
a
b
c
d
```

In [7]:
```
for v in dict.values():
    print(v)
```

```
1
2
3
4
```

In [8]:
```
for k in dict:
    print(dict[k])
```

```
1
2
3
4
```

In [9]:
```
dict_2 = {'a':[1, 11], 'b':[2, 22], 'c':[3, 33], 'd':[4, 44]}
```

In [10]:
```
a = dict_2['c']
```

In [11]:
```
a[1]
```

Out[11]: 33

In [12]:
```
dict_2['c'][1]
```

Out[12]: 33

In [ ]:
```
"""
Write a function birthstate() that:

(1) Takes as input a dictionary of the the birth state for each recent president; and
```

```
(2) Repeats 1) requesting an user to input the name of president, and 2) printing out h
If an user input a wrong name, prints out 'Wrong Name!' and stop the iteration

You should use this dictionary to store the birth state
for each recent president

{'Barack Hussein Obama II': 'Hawaii',
 'George Walker Bush': 'Connecticut',
 'William Jefferson Clinton': 'Arkansas',
 'George Herbert Walker Bush': 'Massachussetts',
 'Ronald Wilson Reagan': 'Illinois',
 'James Earl Carter, Jr': 'Georgia'}


>>> P_dict = {'Barack Hussein Obama II': 'Hawaii',
 'George Walker Bush': 'Connecticut',
 'William Jefferson Clinton': 'Arkansas',
 'George Herbert Walker Bush': 'Massachussetts',
 'Ronald Wilson Reagan': 'Illinois',
 'James Earl Carter, Jr': 'Georgia'}

>>> birthstate(P_dict)
Name of U.S. President: George Walker Bush
Connecticut
Name of U.S. President: William
Wrong Name!
"""
```

In [13]:

```
def birthstate(dict):
    while True:
        name = input('Name of U.S. President: ')
        if name not in dict:
            print('Wrong Name!')
            break
        print(dict[name])
```

In [14]:

```
P_dict = {'Barack Hussein Obama II': 'Hawaii',
 'George Walker Bush': 'Connecticut',
 'William Jefferson Clinton': 'Arkansas',
 'George Herbert Walker Bush': 'Massachussetts',
 'Ronald Wilson Reagan': 'Illinois',
 'James Earl Carter, Jr': 'Georgia'}
```

In [15]:

```
birthstate(P_dict)
```

```
Name of U.S. President: George Walker Bush
Connecticut
Name of U.S. President: William
Wrong Name!
```

In [ ]:

```
"""
Write a function lookup() that:

Implements a phone book lookup application.

Your function takes, as input, a dictionary representing a phone book,
```

```
mapping tuples (containing the first and last name)
to strings (containing phone numbers)

It repeats 1) requesting an user to input first name and last name, and 2) printing out
If an user input nothing, stop the iteration

>>> phonebook = {('Anna', 'Karenina'): '(123) 456-7890',
                 ('Yu', 'Tsun'): '(901) 234-5678',
                 ('Hans', 'Castorp'): '(321) 908-7654'}

>>> lookup(phonebook)
Enter the first name: Anna
Enter the last name: Karenina
(123) 456-7890
Enter the first name:
"""
```

In [16]:

```
def lookup(dict):
    while True:
        fname = input('Enter the first name: ')
        if fname == '':
            break
        lname = input('Enter the last name: ')
        if lname == '':
            break
        kname = (fname, lname)
        print(dict[kname])
```

In [17]:

```
phonebook = {
    ('Anna', 'Karenina'): '(123) 456-7890',
    ('Yu', 'Tsun'): '(901) 234-5678',
    ('Hans', 'Castorp'): '(321) 908-7654'}
```

In [18]:

```
lookup(phonebook)
```

```
Enter the first name: Anna
Enter the last name: Karenina
(123) 456-7890
Enter the first name:
```

In [19]:

```
def lookup(dict):
    while True:
        fname = input('Enter the first name: ')
        lname = input('Enter the last name: ')
        if lname == '' or fname == '':
            break
        kname = (fname, lname)
        print(dict[kname])
```

In [20]:

```
phonebook = {
    ('Anna', 'Karenina'): '(123) 456-7890',
    ('Yu', 'Tsun'): '(901) 234-5678',
    ('Hans', 'Castorp'): '(321) 908-7654'}
```

In [21]:
```
lookup(phonebook)
```

```
Enter the first name: Anna
Enter the last name: Karenina
(123) 456-7890
Enter the first name:
Enter the last name:
```

Random Module

In [ ]:
```
'''
Write a function coin() that:

Returns 'Heads' or 'Tails' with equal probability.

>>> coin()
'Heads'
>>> coin()
'Heads'
>>> coin()
'Tails'
'''
```

In [22]:
```
def coin():
    import random
    t = random.randrange(2)
    if t == 0:
        return 'Heads'
    else:
        return 'Tails'
```

In [23]:
```
coin()
```

Out[23]:
```
'Heads'
```

In [24]:
```
coin()
```

Out[24]:
```
'Tails'
```

In [25]:
```
coin()
```

Out[25]:
```
'Tails'
```

In [ ]:
```
'''
Implement function guess() that:

(1) Takes as input an integer n; and
(2) Implements a simple, interactive number guessing game.

The function should start by choosing a random number
```

```
in the range from 0 up to but not including n.
The function will then repeatedly ask the user to guess
the chosen number; When the user guesses correctly, the function
should print a 'You got it.' message and terminate.
Each time the user guesses incorrectly, the function should
help the user by pringing message 'Too low.', or 'Too high.'.

>>> guess(100)
Enter your guess: 50
Too low.
Enter your guess: 75
Too high.
Enter your guess: 62
Too high.
Enter your guess: 56
Too low.
Enter your guess: 59
Too high.
Enter your guess: 57
You got it!
'''
```

In [26]:

```
def guess(n):
    import random
    t = random.randrange(n)
    while True:
        g = input('Enter your guess: ')
        if int(g) == t:
            print('You got it!')
            break
        elif int(g) > t:
                print('Too high.')
        else:
            print('Too low.')
```

In [27]:

```
guess(100)
```

```
Enter your guess: 50
Too low.
Enter your guess: 75
Too low.
Enter your guess: 90
Too high.
Enter your guess: 85
Too high.
Enter your guess: 80
Too high.
Enter your guess: 79
Too high.
Enter your guess: 78
You got it!
```

In [ ]:

```
'''
Develop function game() that:

(1) Takes integers r and c as input;
(2) Generates a field of r rows and c columns with a bomb at a randomly chosen row and
(3) Then asks users to find the bomb.
```

```
>>> game(2, 3)
Enter next position (format: x y): 1 2
No bomb at position 1 2
Enter next position (format: x y): 1 1
No bomb at position 1 1
Enter next position (format: x y): 2 1
You found the bomb!
'''
```

In [28]:

```
def game(r, c):
    import random
    t_r = random.randrange(1, r + 1)
    t_c = random.randrange(1, c + 1)
    while True:
        guess = input('Enter next position (format: x y): ')
        g_r = int(guess[0])
        g_c = int(guess[-1])
        if t_r == g_r and t_c == g_c:
            print('You found the bomb!')
            break
        else:
            print('No bomb at position {} {}'.format(g_r, g_c))
```

In [29]:

```
game(2, 3)
```

```
Enter next position (format: x y): 1 2
You found the bomb!
```