

# Dictionaries

## Lab07: Scrabble

# Announcements

- Submit all regrade requests by this Friday (03/05)
- Office hours:
  - \* 3:30pm – 5pm (Thursday)
  - \* 2pm – 3:00pm (Friday)

# Lab07: Scrabble Word Finder



Point values for each letter of the alphabet

**Input:** string of letters

**Output:**

- All valid words that can be made using the input letters and their point values
- Output is printed or written to file.  
(See example on the right)

Example run of the program

```
>>> scrabbleWords('buoni')
```

```
obi 5
nub 5
nob 5
nib 5
bun 5
bio 5
bin 5
bi 4
uni 3
ion 3
on 2
nu 2
no 2
in 2
u 1
i 1
```

↑  
Sequence of letters  
"string"  
not word!

Entire English Dictionary  
↓

→ All the words in the file wordlist.txt that can be constructed with the letters 'btoni'

# Break down the problem

1. Create a list of **valid** words that can be made with the given letters: **validWordList**
2. Calculate the point value of each word in **validWordList**
3. Print the word and point values in the desired format

```
>>> scrabbleWords('buoni')
obi      5
nub      5
nob      5
nib      5
bun      5
bio      5
bin      5
bi       4
uni      3
ion      3
on       2
nu       2
no       2
in       2
u        1
i        1
```

The tiles are not relevant  
to solving problem 1.





# Sub problem 1:

## Creating a list of valid words

*wordlist.txt*

aah  
aal  
aalii  
aardvark  
aardwolf  
aasvogel  
aba  
abac  
abaca  
aback  
.....  
.....  
.....



bun  
bi  
bio  
bin  
i  
ion  
nub  
nob  
nib  
nu  
no  
obi  
on  
in  
u  
uni

*filename*

```
L = createWordList('wordlist.txt')
```

```
L: ['aah', 'aal', 'aali', 'aardvark',  
    ..., 'zymotic', 'zymurgy', 'zyzzyva']
```

*List of all the words in 'wordlist.txt'*

*myLetters: 'bouni'*

*Use the function:*

*canWeMakeIT(word, myLetters)*  
*returns True if word can be constructed from my letters*

How can we generate **validWordList** using **L** and **myLetters**? *Discuss with your partner*

**wordlist.txt**

contains all  
the valid words  
in the English  
Dictionary

**validWordList**

List of words in  
wordlist.txt that  
can be made with the  
letters myLetters

# Sub problem 1:

## Create a list of valid words

THIS IS PSEUDO CODE (NOT ALL OF IT IS PYTHON)

Input:

- File containing all valid words (filename)
- string of letters (myLetters)

Output: validWordList

```
L = createWordList(filename)
validWordList = []
for each testWord in L
    if canWeMakeIt(testWord, myLetters)
        append testWord to validWordList
```

# Sub problem 2: Calculate point values

## INPUTS

bun  
bi  
bio  
bin  
i  
ion  
nub  
nob  
nib  
nu  
no  
obi  
on  
in  
u  
uni

What are some possible ways  
of storing in the Scrabble tiles  
in your program

- ① List of named tuples
- ② List of tuples:  $[(\text{'a'}, 1), (\text{'b'}, 3), \dots]$
- ③ Two parallel lists:  $[\text{'a'}, \text{'b'}, \text{'c'}, \dots]$   
point values:  $[1, 3, 3, 2, \dots]$



Scrabble Tiles

key value

- ④ Python Dictionary  $\{\text{'a':1}, \text{'b':3}, \text{'c':3}, \dots\}$

validWordList → letterPoints

Used in lab07: Makes it easy to search for the point value of any letter  
Write and use the helper function:  
getWordPoints(myWord, letterPoints)

## OUTPUT

Point Word List  
[(5, 'bun'),  
(4, 'bi'),  
(5, 'bio'),  
(5, 'bin'),  
(1, 'i'),  
(3, 'ion'),  
(5, 'nub'),  
(5, 'nob'),  
(5, 'nib'),  
(2, 'nu'),  
(2, 'no'),  
(5, 'obi'),  
(2, 'on'),  
(2, 'in'),  
(1, 'u'),  
(3, 'uni')]

# Python Dictionaries

- Used to store a collection of KEY: VALUE pairs
- A KEY maps to a VALUE
- Access each VALUE in the dictionary using the KEY as “index”
- Unlike lists there is no ordering of elements

## Representing Scrabble Tiles in Python:



```
letterPoints = {'a':1, 'b':3,  
'c':3, 'd':2, 'e':1, 'f':4, ...}
```

`letterPoints['a']`

// returns 1

`letterPoints['c']`

// returns 3

# Concept Test

Which of the following is best suited for a dictionary instead of a list?

- A. The order in which people finish a race.
- B. The ingredients necessary for a recipe
- C. The names of world countries and their capital cities
- D. 50 random integers

# Another example

- Let's say we're bird-watching, and we want to keep track of the number of each type of bird we've seen

kind	count
falcon	1
owl	5
hawk	2
eagle	11

- One approach: parallel lists
- The element `kinds[i]` corresponds with `counts[i]`

```
kinds = ['falcon', 'owl', 'hawk', 'eagle']  
counts = [1, 5, 2, 11]
```

# Concep Test:

```
def new_sighting(kinds, counts, sighting):  
    '''(list of str, list of int, str) -> NoneType  
    Add new sighting to parallel lists kinds and counts.  
    '''  
  
    if sighting not in kinds:  
        kinds.append(sighting)  
        ... missing code  
    ind = kinds.index(sighting)  
    counts[ind] = counts[ind] + 1
```

What code should go in place of the missing code?

- A. `counts.append(0)`
- B. `counts.append(1)`
- C. `counts.append(kind)`
- D. No code necessary there

# Dictionaries vs. Parallel Lists

```
bird_dict=  
{ 'falcon': 1, 'owl': 5, 'hawk': 2, 'eagle': 11 }
```

- Rewrite the new\_sighting function
- Compared to parallel lists:
  - Only one dict (not two)
  - No call to index that might search the whole list



# Adding to dictionaries

- Keys must be immutable
- Values can be mutable or immutable
- Use `d[k] = v` to add key `k` with value `v` to dictionary `d`
  - If `k` is already present, its value is overwritten
- To copy all key/value pairs from another dictionary, use the `update` method

# Getting Values from Dictionaries

- Use `d[k]` to obtain the value associated with key `k` of dictionary `d`
- If `k` does not exist, this causes an error
- The `get` method is similar, except it returns `None` instead of giving an error when the key does not exist
- If a second parameter `v` is provided, `get` returns `v` instead of `None` when the key is not found

# Concept Test

What is dictionary `d` created by the following code?

```
d = {3:4}
d[5] = d.get(4, 8)
d[4] = d.get(3, 9)
```

- ▶ A. {3:4, 5:8, 4:9}
- ▶ B. {3:4, 5:8, 4:4}
- ▶ C. {3:4, 5:4, 4:3}
- ▶ D. Error caused by get

# Concept Test

What is dictionary `d` created by the following code?

```
d = {1:5}
d[2] = d.get(1, 6)
d[4] = d.get(3, 7)
```

- ▶ A. {1:5, 2:5, 4:7}
- ▶ B. {1:5, 2:6, 4:7}
- ▶ C. {1:5, 2:1, 4:2}
- ▶ D. Error caused by `get`