Let's Review Some Data Structures

List	Can hold several values	["code", "wiz", "code", "wiz"]
Set	Cannot hold duplicate values	{"code", "wiz"}
2D List	Lists that hold other lists	[["c","o","d","e"], ["w","i","z"]]

All of these require an INDEX to access the value

List	WordsList = ["code", "wiz", "code", "wiz"] 0 1 2 3	WordsList[0] = "code"
Set	WordsSet = {"code", "wiz"} 0 1	WordsSet[0] = "code"
2D List	Words2DList = [["c","o","d","e"], ["w","i","z"]] 0 1 0 1 2 3 0 1 2	Words2DList[0][0] = "c" Words2DList[0][1] = "o" Words2DList[0][2] = "d" Words2DList[0][3] = "e"

Now let's learn DICTIONARIES!

Dictionary	Holds key-value pairs	WordsDict = {"Dog": "Pet", "Treat": "Food"}

Dictionaries **DO NOT** use an **INDEX** to access a value, they use a **KEY**

Dictionary	WordsDict = {"Dog": "Pet", "Treat": "Food"} key value key value	WordsDict["Dog"] = "Pet" WordsDict["Treat"] = "Food"

Why should we use it?

- Look up values **instantly** using a key
- Store **related data** together
- Easily check if something exists

studentGrades = {"Alice": 90, "Bob": 85, "Charlie": 92}	
print(studentGrades["Alice"])	> 90
print(studentGrades["Bob"]) studentGrades["Bob"] = 100 print(studentGrades["Bob"])	> 85 > 100
print(studentGrades["Frank"])	> *ERROR*

Number of occurrences

The text is given in a single line. For each word of the text count the number of its occurrences before it.

Example input
one two one two three two four three

Example output
0 0 1 1 0 2 0 1

Your first instincts might be to keep track the occurrences with variables for each word:

```
oneCount = 0
twoCount = 0
threeCount = 0
fourCount = 0

words = input().split()
for i in range(len(words)):
	word = words[i]
	if word == "one":
		print(twoCount)
		twoCount += 1
	if word == "two":
		print(oneCount)
		oneCount += 1
	if word == "three":
		print(threeCount)
		threeCount += 1
	if word == "four":
		print(fourCount)
		fourCount += 1
```

This solution works, but only for that example. What if the input contains other words, like "seven eight nine ten eleven twelve" **? The solution above can't handle this input!**

Dictionaries let us keep track of **the count of each word** using **Key-Value Pairs.**

```
wordCount = dict()

words = input().split()
for i in range(len(words)):
	word = words[i]
	if word not in wordCount:
		print(0)
		wordCount[word] = 1
	else:
		print(wordCount[word])
		wordCount[word] += 1
```

Synonyms

```
count = int( input() )
synonymsDict = dict()

for i in range(count):
    word1, word2 = input().split()
    synonymsDict[ _____ ] =  _____
    synonymsDict[ _____ ] =  _____

word = input()
print(synonymsDict[ _____ ])
```

Elections

```
count = int( _____ () )
ElectionDict= ____ ()
NameSet = set()

for i in range(count):
    name, votes = input()._____()
    if name not in ElectionDict:
        NameSet.add( name )
        ElectionDict[ _____ ] =  int( _____ )
    else:
        ElectionDict[ _____ ] +=  int( _____ )

for name in sorted(NameSet):
    print(name, ElectionDict[ _____ ])
```

Most Frequent Word

```
count = int( _____ () )
WordDict = ____ ()
WordSet = set()

for i in range(count):
    words = input()._____()
    for word in words:
        if word not in WordDict :
            WordSet.add( word )
            WordDict[ _____ ] = _____
        else:
            WordDict[ _____ ] +=  _____

maxWord, maxValue = "", 0
for word in sorted(WordSet):
    if WordDict[ _____ ] > _____ :
        maxWord, maxValue = _____ , _____

print( _____ )
```

Access Rights

```
files = int( _____ () )
fileSystem = ____ ()

for i in range( files ):
    line = _____().____()
    filename = line[ _ ]

    fileSystem[ _________ ] = []

    if "R" in line:
        fileSystem[ _________ ].append("_______")
    if "W" in line:
        fileSystem[ _________ ].append("_______")
    if "X" in line:
        fileSystem[ _________ ].append("_______")

actions = int( _____ () )

for j in range( actions ):
    action, filename = _____().____()

    if action in fileSystem[ _________ ]:
        print( ___________ )
    else:
        print( ___________ )
```

Countries and Cities

```
countries = int( _____ () )
gpsDict =  ____ ()

for i in range( _________ ):
    line = input().split()
    country = line[ __ ]
    cities = line[ __ : __ ]

    for _____ in cities:
        gpsDict[ _______ ] = _________

cities = int(input())

for j in range( _______ ):
    _______ = input()
    print(gpsDict[ _____ ])
```

Frequency Analysis

```
lines = int( _____ ())
words = ""
for i in range( _____ ):
        words += " " + input()

wordList = words.split()
wordDict = _____ ()

for word in wordList:
        if word not in wordDict:
                wordDict[ _____ ] = __
        else:
                wordDict[ _____ ] += __

FreqList = list()

for word in wordDict:
        # store negative frequency
        FreqList.append( [-1 * wordDict[word], word] )

# default sort: first by -freq, then by word
FreqList = sorted(FreqList)

for item in FreqList:
        # restore original frequency by negating again
        print( item[1] , -1 * item[0])
```

English-Latin Dictionary

```
lines = int( _____ ())

translate = _____ ()

for line in range(lines):
        groups = _____ ()._____ (" - ")

        english = _____ [ __ ]
        latin_words = _____ [ __ ]._____ (", ")

        for latin in latin_words:
                if latin not in translate:
                        translate[latin] = _______
                else:
                        translate[latin] = translate[latin] + ", " + _______

sortedDict = dict(sorted(translate.items()))

print(len(sortedDict))

for latin in sortedDict:
        value = sortedDict[ _____ ]
        print( _____  + " - " + _____ )
```