```
# -*- coding: utf-8 -*-
"""
Created on Wed Sep 21 11:52:34 2016

@author:
WELG
"""

#####################################
# EXAMPLE:  Towers of
Hanoi
#####################################

def printMove(fr, to):
    print('move from ' +
str(fr) + ' to ' + str(to))

def Towers(n, fr, to, spare):
    if n == 1:
        printMove(fr,
to)
    else:
        Towers(n-1, fr, spare, to)
        Towers(1, fr, to, spare)

Towers(n-1, spare, to, fr)

#print(Towers(4, 'P1', 'P2',
'P3'))

#####################################
# EXAMPLE:
fibonacci
#####################################

def fib(x):
    """assumes x an
int >= 0
       returns Fibonacci of x"""
    if x == 0 or x == 1:

return 1
    else:
        return fib(x-1) + fib(x-2)


#####################################
# EXAMPLE:  testing for
palindromes
#####################################

def isPalindrome(s):

    def
toChars(s):
        s = s.lower()
        ans = ''
        for c in s:
            if c in
'abcdefghijklmnopqrstuvwxyz':
                ans = ans + c
        return ans

    def
isPal(s):
        if len(s) <= 1:
            return True
        else:
            return
s[0] == s[-1] and isPal(s[1:-1])

    return
isPal(toChars(s))
```

```
#print(isPalindrome('eve'))
#
#print(isPalindrome('Able was I, ere I saw
Elba'))
#
#print(isPalindrome('Is this a palindrome'))

#####################################
#
EXAMPLE: using dictionaries
#          counting frequencies of words in song
lyrics
#####################################

def lyrics_to_frequencies(lyrics):
    myDict =
{}
    for word in lyrics:
        if word in myDict:
            myDict[word] += 1

else:
            myDict[word] = 1
    return myDict


she_loves_you = ['she', 'loves',
'you', 'yeah', 'yeah',
'yeah','she', 'loves', 'you', 'yeah', 'yeah', 'yeah',
'she', 'loves',
'you', 'yeah', 'yeah', 'yeah',

'you', 'think', "you've", 'lost', 'your',
'love',
'well', 'i', 'saw', 'her', 'yesterday-yi-yay',
"it's", 'you',
"she's", 'thinking', 'of',
'and', 'she', 'told', 'me', 'what', 'to',
'say-yi-yay',

'she', 'says', 'she', 'loves', 'you',
'and', 'you', 'know', 'that',
"can't", 'be', 'bad',
'yes', 'she', 'loves', 'you',
'and', 'you', 'know', 'you',
'should', 'be', 'glad',

'she', 'said', 'you', 'hurt', 'her', 'so',
'she', 'almost', 'lost',
'her', 'mind',
'and', 'now', 'she', 'says', 'she', 'knows',
"you're", 'not', 'the',
'hurting', 'kind',

'she', 'says', 'she', 'loves', 'you',
'and', 'you', 'know', 'that',
"can't", 'be', 'bad',
'yes', 'she', 'loves', 'you',
'and', 'you', 'know', 'you',
'should', 'be', 'glad',

'oo', 'she', 'loves', 'you', 'yeah', 'yeah', 'yeah',
'she', 'loves',
'you', 'yeah', 'yeah', 'yeah',
'with', 'a', 'love', 'like', 'that',
'you', 'know', 'you',
'should', 'be', 'glad',

'you', 'know', "it's", 'up', 'to', 'you',
'i', 'think',
"it's", 'only', 'fair',
'pride', 'can', 'hurt', 'you', 'too',
'pologize', 'to',
'her',
```

```
'Because', 'she', 'loves', 'you',
'and', 'you', 'know', 'that', "can't",
'be', 'bad',
'Yes', 'she', 'loves', 'you',
'and', 'you', 'know', 'you', 'should', 'be',
'glad',

'oo', 'she', 'loves', 'you', 'yeah', 'yeah', 'yeah',
'she', 'loves', 'you', 'yeah',
'yeah', 'yeah',
'with', 'a', 'love', 'like', 'that',
'you', 'know', 'you', 'should', 'be',
'glad',
'with', 'a', 'love', 'like', 'that',
'you', 'know', 'you', 'should', 'be',
'glad',
'with', 'a', 'love', 'like', 'that',
'you', 'know', 'you', 'should', 'be',
'glad',
'yeah', 'yeah', 'yeah',
'yeah', 'yeah', 'yeah', 'yeah'
]

beatles =
lyrics_to_frequencies(she_loves_you)


def most_common_words(freqs):
    values =
freqs.values()
    best = max(freqs.values())
    words = []
    for k in freqs:
        if
freqs[k] == best:
            words.append(k)
    return (words, best)

def
words_often(freqs, minTimes):
    result = []
    done = False
    while not done:
        temp
= most_common_words(freqs)
        if temp[1] >= minTimes:
            result.append(temp)

          for w in temp[0]:
                del(freqs[w])  #remove word from dict

else:
            done = True
    return result

#print(words_often(beatles,
5))

#######################################
# EXAMPLE: comparing fibonacci using
memoization
#######################################


def fib(n):
    if n == 1:
        return
1
    elif n == 2:
        return 2
    else:
        return fib(n-1) + fib(n-2)


def
```

```
fib_efficient(n, d):
    if n in d:
        return d[n]
    else:
        ans =
fib_efficient(n-1, d)+fib_efficient(n-2, d)
        d[n] = ans
        return ans

d =
{1:1, 2:2}

argToUse = 34
#print("")
#print('using
fib')
#print(fib(argToUse))
#print("")
#print('using
fib_efficient')
#print(fib_efficient(argToUse, d))
```