IMPORTANT FACTS (not a question):   Consider the following code:

```
def f(x):
    return (x + 1) * (x + 2)
def g(x, y):
    return x + y
print (g(f(5), f(3)))
```

It calls **f(5)** which returns **42**, then calls **f(3)** which returns **20**,
then calls **g(42,20)** which returns **62**, which is printed.

This may be obvious, but you should remember: When calling a function, the
interpreter does the following things, in sequence:
    First: evaluate each of the function's arguments, left to right.
    Second: call the function with the values obtained in the First step.

1. Consider this program:

```
def f(x, y):
    print ("f", end="")        ← don't miss this print statement!
    return (x > y)
def g(x, y):
    print ("g", end="")
    return (x > 2*y)
print (f(1, 0) and g(-1, -2) and f(3, 4) and g(5, 6))
```

  a) list the functions, with their arguments, in the order they are called

  b) what is printed?

2. Consider this program:

```
def f( x, y ):
    return((type(x) == type(y) == int) and
            (100 > x > y > 0) and
            (x/10 + x%10 + y/10 + y%10 < 5) and
            (x % 10 == y % 10) and
            (x + y == 22)
```

Find one pair of values for **x** and **y** that make **f(x,y)** return **True**.

3. Here is a program, and a data file:
   What does the program print?

Program `names.py`:	File `de_numbers.txt`:
| ```python
file = open('de_numbers.txt', 'r')
lines = file.readlines()

ger_num = dict()
for line in lines:
    line = line.rstrip('\n\r')
    (eng, ger) = line.split()
    ger_num[eng] = ger

eng_num = dict()
for x in ger_num:
    ger = ger_num[x]
    eng_num[ger] = x

for x in sorted(eng_num.keys()):
    print (eng_num[x], end='')

print()
``` | ```
one    ein
two    zwei
three drei
four   vier
five   funf
six    sechs
``` |

4. Here is a program, and a data file. What does it output?

Program `counts.py`:	File `words.txt`:
| ```python
file = open('words.txt', 'r')
lines = file.readlines()
words = []
for line in lines:
    line = line.rstrip('\n\r')
    for x in line.split():
        words.append(x)

counts = dict()
for word in words:
    if word not in counts:
        counts[word] = 0
    counts[word] += 1

for k in sorted(counts.keys()):
    print (k, counts[k])
``` | ```
apple cherry banana apple
plum cherry banana
apple orange apple
banana plum orange
papaya
``` |