Work with your neighbor. (This will be graded for participation only.)

1. Write a function `min_max(L)` that takes a list `L` of integers and returns a tuple of the smallest and largest **EVEN** numbers in `L`. You may use the built-in min and max functions.

ANS:

```
def min_max(L):

    evens = []

    #first get a list of the even numbers

    for num in L:

        if num % 2 == 0:

            evens.append(num)

    return (min(evens), max(evens))
```

Note: the parentheses in the return statement are optional. It could be written like this:

```
    return min(evens), max(evens)
```

2. Given the dictionary `catalog` shown below:

```
>>> catalog
{ 'MIS': {'mis 101': 4, 'mis 102': 3, 'mis 202': 2},
  'CSC': {'csc 110': 4, 'csc 120': 4, 'csc 352': 3},
  'ECE': {'ece 111': 3, 'ece 222': 3, 'ece 333': 4}}
```

Use the `items()` method in your solutions for a) and b) below:

a) Print the keys and values of the `catalog`, separated by a ":"

ANS:

```
for key, value in catalog.items():

        print(key, ':', value)
```

b) Print the keys and value of `catalog`, separated by a ":", in sorted order of the keys.

ANS:

```
for key, value in sorted(catalog.items()):

        print(key, ':', value)
```

3. Given these assignments:

```
v = "ABCDEFGHIJ"
w = ( ('aa','ab','bc'), (12, 23, 34), [45, 56, 67, 78] )
x = { 'abc' : 12, 'cde' : 34, 'efg' : 56 }
y = [ ['pqr', 'stu','abc','def'], ['uvw','xyz','bcd','cde']  ]
```

   (a) For each of the variables `v, w, x, y,` indicate what kind of a data object its value is (i.e., number, list, dictionary, etc.).

   ANS:
   v is a string
   w is a tuple
   x is a dictionary
   y is a list

   (b) Does this give an error?
   ```
            w[2][3] = "hello"
   ```
   ANS:
   No, because `w[2]` is a list and `w[2][3]` assigns to its last element. The contents of the tuple `w` would then be the following:

   ```
   ( ('aa', 'ab','bc'), (12, 23, 34), [45, 56, 67, 'hello'] )
   ```

   (c) Does this give an error?

   ```
            w[1][2] = "sunshine"
   ```
   ANS:
   Yes, because `w[1]` is a tuple and `w[1][2]` attempts to assign to a tuple element.

   (d) What are the contents of `x` after the following statement:

   ```
          x[ v[2:4].lower() ] = w[1][2]
   ```
   ANS:

   ```
   x = { 'abc' : 12, 'cde' : 34, 'efg' : 56 , 'cd' : 34}
   ```

   (Notice that the assignment added a new key/value pair to the dictionary.)

NOTE: We didn't get to problem 3. It has been moved to ICA-7.