Dictionaries and lists each have their strengths and weaknesses. Lists keep data ordered, but you need to search the entire list to find a particular piece of data. Dictionaries allow fast access to data based upon a key, but the data is not ordered. When storing data, you need to decide which structure is better.

There are some times when it makes the most sense to use a list:

1. If the data must be ordered.
2. If there is no unique key that could be used to identify the individual data items.
3. If you potentially would use a lot of different keys to access the data.

There are other times when it makes the most sense to use a dictionary:

1. If there is a unique key that identifies the individual data items which is used to access those data items.
2. If that key is frequently used to access the data items, which would result in a lot of searching if you used a list.

Note that these are not the only considerations. They are just examples of the things that you want to think about when selecting an appropriate data structure.

In some cases, it makes the most sense to use lists in some parts of your program and dictionaries in others to store the same data. In that case, you may consider converting back and forth between dictionaries and lists. If that is the case, you likely will want to make sure that the items that are used as the keys in the dictionary form are stored within the items of the list. Otherwise, you will not be able to convert from a list to a dictionary and when you convert from a dictionary to a list the information in the keys would be lost.

Consider the following simple example where we are storing information about students and their grade point averages. We could store that data as a list of tuples:

```
[("Tamika Barker", 3.9), ("Elmer Brasher", 2.8), ("Raymond Hathaway", 3.3), ("Rebekah Bailey", 3.5)]
```

In this form, the data could easily be sorted by GPA, but to find a particular student's GPA you would need to search the entire list.

In contrast, we could store that data as a dictionary:

```
{"Tamika Barker": 3.9, "Elmer Brasher": 2.8, "Raymond Hathaway": 3.3, "Rebekah Bailey": 3.5}
```

In this form, it is easy to find the GPA for a particular student, but we can't sort it by GPA. If we sometimes want the data in dictionary form and sometimes want it in list form, it might make sense to store the entire tuple as the value in the dictionary. This will make the conversion slightly easier (with more complex data storing data in the same format within the list and as values in the dictionary can really pay off, whereas here it is pretty easy to convert back and forth).

If we wanted to convert from the list form to the dictionary form, we could write simple code like this:

```
def convert_list2dict(gpalist):
    """
    Input: gpalist - list of (student name, gpa) tuples
    Output: a dictionary mapping student names to their gpa
    """
    result = {}
    for item in gpalist:
        result[item[0]] = item[1]
    return result
```

Note that if it is possible for the same student to appear twice in the list, you would have to check if that happens and take care of it appropriately. It is left as an exercise for you to create the opposite function `convert_dict2list(gpadict)`.

Mark as completed