CS5001 Spring 2022
Homework 6: Nested Lists, Dictionaries, and JSON files

Q0. Write a recursive Python function called flatten_to_dict(lst) which takes one parameter, a (possibly nested) list. If the parameter is *not* a list, raise TypeError. Each element of the list should be either be a dictionary or another (possibly nested) list. If a list element is anything else, it should simply be skipped. Your mission is to combined all the dictionaries found, at whatever levels of nesting, into a single dictionary, and return that dictionary. If any of the dictionaries would result in ***duplicate keys with different values***, then the final value for that key should be a ***list*** of ***all*** of the values encountered for that key. Your file should include a minimum of six (6) test cases in its main() function; use our standard template to ensure that main() only runs when intended. Include a documentation string as part of your function definition.

Here are some examples:

```
    >>> l1 = [{'a': 'alpha', 'b': 'bravo'}, [[{'c': 'charlie'}],
              [{'e': 'echo'}]],  {'d': 'delta'}]
    >>> d1 = flatten _to_dict(l1)
    {'a': 'alpha', 'b': 'bravo', 'c': 'charlie', 'e': 'echo', 'd': 'delta'}

    >>> l2 = [1, ('a' 'b'), {'c': 'charlie'}]
    >>> d2 = flatten_to_dict(l2)
    {'c': 'charlie'}

    >>> l3 = [{'a': 'able'}, {'b': 'baker'}, "Ignore", {'b': 'bravo'}]
    >>> d3 = flatten_to_dict(l3)
   {'a': 'able', 'b': ['baker', 'bravo']}

    >>>> flatten_to_dict(3.14)
    TypeError: lst should be a list
```

Q1. Write a *separate* program, in its own .py file, to **save** your resulting dictionary from **Q0** to a .json file. (Json format is particularly well-suited to saving Python dictionaries.) Import the standard, built-in json package; no installation is required. The name of your output file is up to you, so long as it ends in .json.

The following construct is the preferred way to write a dictionary to a .json file:

```
    with open(filename, "x") as outfile:
        json.dump(mydict, outfile)
```

Wrap it in a try-except block that handles the 3 most common file I/O exceptions. Your function should return Boolean indicating whether your json file was successfully created. Include a doc string for your function and a main() that creates 3 test files with different names and dictionaries: an empty dictionary, a small dictionary, and a dictionary illustrating a list value for one key, resulting from merging multiple occurrences of that key. See https://realpython.com/python-json/#a-real-world-example-sort-of for more about json.

Please upload all relevant files, including your sample .json output files, to Canvas.