Exercises: Dictionaries and Sets

These exercises are taken from Chapter 8, Building Python Programs, A Back to Basics Approach by Stuart Reges, Marty Stepp, and Allison Obourn. **ISBN-10:** 0135205980, **ISBN-13:** 978-0135205983.

1. Write a function called `intersect` that accepts two dictionaries whose keys are strings and whose values are integers as parameters and returns a new dictionary containing only the key/value pairs that exist in both of the dictionaries. In order for a key/value pair to be included in your result, not only do both dictionaries need to contain a mapping for that key, but they need to map it to the same value. For example, if the two dictionaries passed are:

   ```
   {"Janet": 87, "Logan": 62, "Whitaker": 46, "Alyssa": 100, "Stef": 80,
   "Jeff": 88, "Kim": 52, "Sylvia": 95}

   {"Logan": 62, "Kim": 52, "Whitaker": 52, "Jeff": 88, "Stef": 80, "Brian":
   60, "Lisa": 83, "Sylvia": 87}
   ```

   your function would return a new dictionary (the order of the key/value pairs does not matter). Using the dictionaries above, output from your program should look like this:

   ```
   Intersection: {"Logan": 62, "Stef": 80, "Jeff": 88, "Kim": 52}
   ```

2. Write a function called `is_1_to_1` that accepts a dictionary whose keys and values are strings as its parameter and returns `True` if no two keys map to the same value. For example, for the following dictionary your function should return `False` because both Hawking and Newton map to the same value:

   ```
   {"Marty": "206-9024", "Hawking": "123-4567",
    "Smith": "949-0504", "Newton": "123-4567"}
   ```

   but for the following dictionary your function should return `True` because each key maps to a unique value:

   ```
   {"Marty": "206-9024", "Hawking": "555-1234",
    "Smith": "949-0504", "Newton": "123-4567"}
   ```

   The empty dictionary is considered 1-to-1 and returns `True`.

   Using the dictionaries above and an empty dictionary, output from your program should look like this:

   ```
   Dictionary:  {'Marty': '206-9024', 'Hawking': '123-4567', 'Smith': '949-
   0504', 'Newton': '123-4567'} does not have unique entries.

   Dictionary:  {'Marty': '206-9024', 'Hawking': '555-1234', 'Smith': '949-
   0504', 'Newton': '123-4567'} has unique entries.

   Dictionary:  {} has unique entries.
   ```

3. Write a function called `max_occurrences` that accepts a list of integers as a parameter and returns a key, value pair which are: the most frequently occurring integer and the number of times the most frequently occurring integer (the "mode") occurs in the list. Solve this problem using a dictionary as auxiliary storage. If the list is empty, return `(0,0)`.

   Your program should:
   i) Generate a list of 100 integers in length containing numbers between 0-9.
   ii) Print the list
   iii) Print the number that occurred most frequently and the mode.

   The output from your program should look like this:

```
The list is:  [9, 7, 6, 2, 7, 6, 9, 4, 2, 8, 5, 5, 3, 7, 6, 3, 0, 2, 2, 2,
7, 3, 6, 5, 2, 1, 2, 6, 8, 0, 5, 4, 1, 9, 1, 0, 6, 5, 1, 8, 0, 5, 2, 2, 0,
3, 0, 4, 8, 8, 3, 3, 8, 6, 8, 6, 6, 9, 2, 7, 9, 1, 0, 6, 1, 5, 4, 8, 7, 0,
7, 6, 4, 3, 6, 7, 9, 4, 9, 6, 4, 4, 0, 0, 7, 2, 2, 2, 1, 4, 1, 5, 5, 3, 0,
9, 5, 3, 1, 4]


Number 6 appears most frequently - 13  times.
```

4. Write a function called `has_odd` that accepts a set of integers as a parameter and returns `True` if the set contains at least one odd integer and `False` otherwise. If passed the empty set, your function should return `False`.

5. Write a function called `symmetric_set_difference` that accepts two sets as parameters and returns a new set containing their symmetric set difference (that is, the set of elements contained in either of the two sets but not in both). For example, the symmetric difference between the sets `{1, 4, 7, 9}` and `{2, 4, 5, 6, 7}` is `{1, 2, 5, 6, 9}`.

Further problems can be found at:

Code step by step

Codingbat Python

Note: you do not have to create accounts on these websites if you do not wish to.