20 Problems

Here are 20 Python problems related to working with lists, dictionaries, sets, and copying collections. Each problem comes with a solution.

Problems:

1. **List Sum:**
   - Write a function that takes a list of numbers and returns their sum.

2. **Duplicate Removal:**
   - Write a function to remove duplicates from a list.

3. **List Reversal:**
   - Write a function to reverse a list.

4. **List Intersection:**
   - Write a function that takes two lists and returns a list containing their common elements.

5. **Set Operations:**
   - Write a function that performs union, intersection, and difference operations on two sets.

6. **Dictionary Merging:**
   - Write a function that merges two dictionaries.

7. **Dictionary Filtering:**
   - Write a function that removes items with a specific value from a dictionary.

8. **Tuple Average:**
   - Write a function that calculates the average of elements in a tuple.

9. **Nested List Flattening:**
   - Write a function that flattens a nested list.

10. **List Sorting:**
    - Write a function that takes a list of dictionaries and sorts them based on a specific key.

11. **Set Symmetric Difference:**
    - Write a function that calculates the symmetric difference between two sets.

12. **Dictionary Key Extraction:**
    - Write a function that extracts keys from a list of dictionaries.

13. **Dictionary Value Update:**
    - Write a function that updates the values of a specific key in a list of dictionaries.

14. **List Element Frequency:**
    - Write a function that counts the frequency of each element in a list.

15. **List Chunking:**
    - Write a function that divides a list into chunks of a specified size.

16. **List Rotation:**
    - Write a function that rotates a list by a given number of positions.

17. **Set Subset Check:**
    - Write a function that checks if one set is a subset of another.

18. **Dictionary Key Filtering:**
    - Write a function that removes keys from a dictionary based on a specified condition.

19. **Tuple Sorting:**
    - Write a function that sorts a list of tuples based on the second element of each tuple.

20. **Deep Copy Verification:**
    - Write a function that demonstrates the difference between shallow copy and deep copy in Python.

 # Solutions:

1. **List Sum:**
   ```python
   def list_sum(numbers):
       return sum(numbers)
   ```

2. **Duplicate Removal:**
   ```python
   def remove_duplicates(lst):
       return list(set(lst))
   ```

3. **List Reversal:**
   ```python
   def reverse_list(lst):
       return lst[::-1]
   ```

4. **List Intersection:**
   ```python
   def list_intersection(list1, list2):
       return list(set(list1) & set(list2))
   ```

5. **Set Operations:**
   ```python
   def set_operations(set1, set2):
   ```

```
    union_set = set1 | set2
    intersection_set = set1 & set2
    difference_set = set1 - set2
    return union_set, intersection_set, difference_set
```

6. **Dictionary Merging:**
```python
def merge_dicts(dict1, dict2):
    merged_dict = dict1.copy()
    merged_dict.update(dict2)
    return merged_dict
```

7. **Dictionary Filtering:**
```python
def filter_dict(input_dict, value_to_remove):
    return {key: value for key, value in input_dict.items() if value != value_to_remove}
```

8. **Tuple Average:**
```python
def tuple_average(tup):
    return sum(tup) / len(tup)
```

9. **Nested List Flattening:**
```python
def flatten_nested_list(nested_list):
    return [item for sublist in nested_list for item in sublist]
```

10. **List Sorting:**
```python
def sort_list_of_dicts(list_of_dicts, key):
    return sorted(list_of_dicts, key=lambda x: x[key])
```

Certainly! Here are solutions for the remaining problems:

11. **Set Symmetric Difference:**
```python
def symmetric_difference(set1, set2):
    return set1.symmetric_difference(set2)
```

12. **Dictionary Key Extraction:**
```python
def extract_keys(list_of_dicts):
```

```
    return [key for dct in list_of_dicts for key in dct.keys()]
```

13. **Dictionary Value Update:**
```python
def update_values(list_of_dicts, key_to_update, new_value):
    for dct in list_of_dicts:
        if key_to_update in dct:
            dct[key_to_update] = new_value
    return list_of_dicts
```

14. **List Element Frequency:**
```python
def element_frequency(lst):
    freq_dict = {}
    for element in lst:
        freq_dict[element] = freq_dict.get(element, 0) + 1
    return freq_dict
```

15. **List Chunking:**
```python
def chunk_list(lst, chunk_size):
    return [lst[i:i + chunk_size] for i in range(0, len(lst), chunk_size)]
```

16. **List Rotation:**
```python
def rotate_list(lst, positions):
    positions %= len(lst)
    return lst[positions:] + lst[:positions]
```

17. **Set Subset Check:**
```python
def is_subset(set1, set2):
    return set1.issubset(set2)
```

18. **Dictionary Key Filtering:**
```python
def filter_keys(input_dict, condition):
    return {key: value for key, value in input_dict.items() if condition(key)}
```

19. **Tuple Sorting:**
```python
def sort_tuples_by_second_element(list_of_tuples):
```

```
    return sorted(list_of_tuples, key=lambda x: x[1])
```

20. **Deep Copy Verification:**
```python
import copy

original_list = [1, [2, 3], 4]
shallow_copied_list = copy.copy(original_list)
deep_copied_list = copy.deepcopy(original_list)

# Modify the original list
original_list[1][0] = 'x'

print(original_list)            # Output: [1, ['x', 3], 4]
print(shallow_copied_list)      # Output: [1, ['x', 3], 4]
print(deep_copied_list)         # Output: [1, [2, 3], 4]
```