10 Assignments

Assignment 1: List Operations

Instructions:
- Declare a variable `numbers` with a list of integers (e.g., [4, 2, 7, 1, 9]).
- Write a function `sum_of_elements` that takes the list as an argument and returns the sum of its elements.

Solution:
```python
# Assignment 1 Solution
numbers = [4, 2, 7, 1, 9]

def sum_of_elements(lst):
    return sum(lst)

result = sum_of_elements(numbers)
print(result)  # Output: 23
```

Assignment 2: Set Operations

Instructions:
- Declare two variables, `set1` and `set2`, with sets of your choice.
- Write a function `common_elements` that takes both sets as arguments and returns a set containing their common elements.

Solution:
```python
# Assignment 2 Solution
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}

def common_elements(s1, s2):
    return s1.intersection(s2)

result = common_elements(set1, set2)
print(result)  # Output: {3, 4}
```

Assignment 3: Dictionary Operations

Instructions:
- Declare a variable `my_dict` with a dictionary of your choice.
- Write a function `dict_key_count` that takes the dictionary as an argument and returns the count of keys in the dictionary.

Solution:
```python
# Assignment 3 Solution
my_dict = {'apple': 3, 'banana': 5, 'orange': 2}

def dict_key_count(dct):
    return len(dct.keys())

result = dict_key_count(my_dict)
print(result)  # Output: 3
```

Assignment 4: List Sorting

Instructions:
- Declare a variable `students` with a list of dictionaries, each containing 'name' and 'score'.
- Write a function `sort_students_by_score` that takes the list as an argument and returns the list sorted by 'score' in descending order.

Solution:
```python
# Assignment 4 Solution
students = [{'name': 'Alice', 'score': 85},
            {'name': 'Bob', 'score': 92},
            {'name': 'Charlie', 'score': 78}]

def sort_students_by_score(lst):
    return sorted(lst, key=lambda x: x['score'], reverse=True)

result = sort_students_by_score(students)
print(result)
# Output: [{'name': 'Bob', 'score': 92}, {'name': 'Alice', 'score': 85}, {'name': 'Charlie', 'score': 78}]
```

Assignment 5: Tuple Operations

Instructions:
- Declare a variable `coordinates` with a tuple of (x, y) values.
- Write a function `distance_from_origin` that takes the tuple as an argument and returns the Euclidean distance from the origin.

Solution:
```python
# Assignment 5 Solution
coordinates = (3, 4)

def distance_from_origin(coord):
    return (coord[0]**2 + coord[1]**2)**0.5
```

```
result = distance_from_origin(coordinates)
print(result)  # Output: 5.0
```

Assignment 6: List Manipulation

Instructions:
- Declare a variable `word_list` with a list of words.
- Write a function `filter_long_words` that takes the list and an integer `n` as arguments and returns a new list with words longer than `n`.

Solution:
```python
# Assignment 6 Solution
word_list = ['apple', 'banana', 'orange', 'kiwi']

def filter_long_words(lst, n):
    return [word for word in lst if len(word) > n]

result = filter_long_words(word_list, 5)
print(result)  # Output: ['banana', 'orange']
```

Assignment 7: Set Operations

Instructions:
- Declare two variables, `set_a` and `set_b`, with sets of your choice.
- Write a function `union_and_difference` that takes both sets as arguments and returns a tuple containing their union and set difference.

Solution:
```python
# Assignment 7 Solution
set_a = {1, 2, 3, 4}
set_b = {3, 4, 5, 6}

def union_and_difference(s1, s2):
    union_set = s1.union(s2)
    difference_set = s1 - s2
    return union_set, difference_set

result = union_and_difference(set_a, set_b)
print(result)  # Output: ({1, 2, 3, 4, 5, 6}, {1, 2})
```

Assignment 8: Deep Copy Verification

Instructions:

- Declare a variable `original_list` with a list containing nested elements.
- Create a shallow copy and a deep copy of `original_list`. Modify the original list and observe the changes in the copies.

Solution:
```python
# Assignment 8 Solution
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]
```

 Assignment 9: Dictionary Filtering

Instructions:
- Declare a variable `my_dict` with a dictionary of your choice.
- Write a function `filter_dictionary` that takes the dictionary and a value as arguments and returns a new dictionary without key-value pairs with that value.

Solution:
```python
# Assignment 9 Solution
my_dict = {'apple': 3, 'banana': 5, 'orange': 2}

def filter_dictionary(dct, value_to_remove):
    return {key: value for key, value in dct.items() if value != value_to_remove}

result = filter_dictionary(my_dict, 5)
print(result)  # Output: {'apple': 3, 'orange': 2}
```

 Assignment 10: List Chunking

Instructions:
- Declare a variable `data` with a list of elements.
- Write a function `chunk_data` that takes the list and a chunk size as arguments and returns a list of chunks.

Solution:
```python
```

```
# Assignment 10 Solution
data = [1, 2, 3, 4, 5, 6, 7, 8, 9]

def chunk_data(lst, chunk_size):
    return [lst[i:i + chunk_size] for i in range(0, len(lst), chunk_size)]

result = chunk_data(data, 3)
print(result)  # Output: [[1, 2, 3
```