Exercises

1. Write a Python function that takes a list of integers as input and returns a new list containing only the even numbers.
2. Write a Python function that takes two dictionaries as input and returns a new dictionary that contains only the key-value pairs that are present in both input dictionaries.
3. Write a Python function that takes a string as input and returns the most common letter in the string.
4. Write a Python function that takes a list of tuples as input, where each tuple contains a name and an age, and returns a list of names of people who are over a certain age.
5. Write a Python function that takes a list of numbers as input and returns the two numbers in the list that add up to a specific target.
6. Write a Python function that takes a list of strings as input and returns a new list that contains only the strings that have at least one uppercase letter.
7. Write a Python function that takes a list of integers as input and returns a new list that contains the differences between adjacent elements in the input list.
8. Write a Python class Circle that represents a circle with a given radius. The class should have methods to calculate the circle's area and circumference.
9. Write a Python function that takes a list of dictionaries as input, where each dictionary represents a person and has keys 'name' and 'age', and returns a new list of names sorted by age in ascending order.
10. Write a Python function that takes a list of integers as input and returns a new list that contains only the elements that appear more than once in the input list.

Exercises and Solution

1. Write a Python function that takes a list of integers as input and returns a new list containing only the even numbers.

```
def even_numbers(lst):
    return [num for num in lst if num % 2 == 0]
# example usage:
nums = [1, 2, 3, 4, 5, 6, 7, 8]
even_nums = even_numbers(nums)
print(even_nums)  # Output: [2, 4, 6, 8]
```

2. Write a Python function that takes two dictionaries as input and returns a new dictionary that contains only the key-value pairs that are present in both input dictionaries.

```
def intersect_dicts(dict1, dict2):
    return {key: value for key, value in dict1.items() if key in dict2 and dict2[key] == value}
```

```
# example usage:
dict1 = {'a': 1, 'b': 2, 'c': 3}
dict2 = {'a': 1, 'b': 3, 'd': 4}
intersected = intersect_dicts(dict1, dict2)
print(intersected)  # Output: {'a': 1}
```

3. Write a Python function that takes a string as input and returns the most common letter in the string.

```
def most_common_letter(string):
    letter_counts = {}
    for letter in string:
        if letter not in letter_counts:
            letter_counts[letter] = 1
        else:
            letter_counts[letter] += 1
    most_common = max(letter_counts, key=letter_counts.get)
    return most_common
# example usage:
text = "The quick brown fox jumps over the lazy dog"
common_letter = most_common_letter(text)
print(common_letter)  # Output: 'o'
```

4. Write a Python function that takes a list of tuples as input, where each tuple contains a name and an age, and returns a list of names of people who are over a certain age.

```
def over_age(name_age_list, age):
    return [name for name, age_ in name_age_list if age_ > age]


# example usage:
people = [('Alice', 25), ('Bob', 35), ('Charlie', 20), ('David', 40)]
over_30 = over_age(people, 30)
print(over_30)  # Output: ['Bob', 'David']
```

5. Write a Python function that takes a list of numbers as input and returns the two numbers in the list that add up to a specific target.

```
def two_sum(nums, target):
    num_dict = {}
    for i, num in enumerate(nums):
        complement = target - num
        if complement in num_dict:
            return [num_dict[complement], i]
        num_dict[num] = i

# example usage:
nums = [2, 7, 11, 15]
target = 9
result = two_sum(nums, target)
print(result)  # Output: [0, 1]
```

6. Write a Python function that takes a list of strings as input and returns a new list that contains only the strings that have at least one uppercase letter.

```
def uppercase_strings(lst):
    return [string for string in lst if any(letter.isupper() for letter in string)]

# example usage:
strings = ['hello', 'WORLD', 'Python', 'is', 'FUN']
uppercase_strings = uppercase_strings(strings)
print(uppercase_strings)  # Output: ['WORLD', 'Python', 'FUN']
```

7. Write a Python function that takes a list of integers as input and returns a new list that contains the differences between adjacent elements in the input list.

```
def adjacent_differences(lst):
    return [lst[i+1] - lst[i] for i in range(len(lst)-1)]
# example usage:
nums = [3, 6, 9, 12, 15]
differences = adjacent_differences(nums)
print(differences)  # Output: [3, 3, 3, 3]
```

8. Write a Python class Circle that represents a circle with a given radius. The class should have methods to calculate the circle's area and circumference.

```
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14159 * self.radius ** 2

    def circumference(self):
        return 2 * 3.14159 * self.radius

# example usage:
my_circle = Circle(5)
print(my_circle.area())  # Output: 78.53975
print(my_circle.circumference())  # Output: 31.4159
```

9. Write a Python function that takes a list of dictionaries as input, where each dictionary represents a person and has keys 'name' and 'age', and returns a new list of names sorted by age in ascending order.

```
def sort_names_by_age(people):
    return [person['name'] for person in sorted(people, key=lambda x: x['age'])]

# example usage:
people = [{'name': 'Alice', 'age': 25}, {'name': 'Bob', 'age': 35}, {'name': 'Charlie', 'age': 20}]
sorted_names = sort_names_by_age(people)
print(sorted_names)  # Output: ['Charlie', 'Alice', 'Bob']
```

10. Write a Python function that takes a list of integers as input and returns a new list that contains only the elements that appear more than once in the input list.

```
def repeated_elements(lst):
    return list(set([num for num in lst if lst.count(num) > 1]))

# example usage:
nums = [1, 2, 3, 2, 4, 3, 5, 6, 5]
repeated = repeated_elements(nums)
print(repeated)  # Output: [2, 3, 5]
```