Exercises on Basic Python, Object-Oriented Programming, and Testing

Basic Python (Chapter 2)

1. Compute the result and explain how Python evaluates the expressions below:
   - `5 + 3 * 2`
   - `10 / 3`
   - `10 // 3`
   - `10 % 3`
   - `5 ** 2`
2. What will be the output of the following Python program? Explain why.

```
x = 3.5
y = int(x)
print(y, type(y))
```

3. Lists:
   (a) Create a list with the first five prime numbers.
   Example: Output: `[2, 3, 5, 7, 11]`
   (b) Write a function to remove all even numbers from a given list.
   Example:
   Input: `[1,2,3,4,5,6]`
   Output: `[1,3,5]`
   (c) Implement a function that finds the second largest number in a list.
   Example:
   Input: `[10,20,4,45,99]`
   Output: `45`
   (d) Write a function that flattens a nested list.
   Example:
   Input: `[[1,2],[3,4],[5]]`
   Output: `[1,2,3,4,5]`
   (e) Write a function that rotates a list to the right by a given number of positions.
   Example:
   Input: `[1,2,3,4,5]`, Rotate by 2
   Output: `[4,5,1,2,3]`
   (f) Write a function to count the occurrences of each element in a list.
   Example:
   Input: `[1,1,2,3,3,3,4]`
   Output: {`1:2, 2:1, 3:3, 4:1`}
4. Tuples:
   (a) Convert a list of tuples into a dictionary.
   Example:
   Input: `[(1,'a'), (2,'b'), (3,'c')]`
   Output: {`1:'a', 2:'b', 3:'c'`}

(b) Write a function that swaps the first and last elements of a tuple.
Example:
Input: `(1,2,3,4,5)`
Output: `(5,2,3,4,1)`

(c) Write a function that finds the maximum and minimum values in a tuple.
Example:
Input: `(4, 7, 1, 9)`
Output: `(9,1)`

(d) Write a function that converts a tuple of numbers into a single concatenated string.
Example:
Input: `(1,2,3,4)`
Output: `"1234"`

5. Dictionaries

(a) Write a function that merges two dictionaries, summing values of common keys.
Example: Input: `{'a':1, 'b':2}`, `{'b':3, 'c':4}`
Output: `{'a':1, 'b':5, 'c':4}`

(b) Write a function that inverts a dictionary (keys become values and vice versa).
Example: Input: `{'a': 1, 'b': 2}`
Output: `{1: 'a', 2: 'b'}`

(c) Write a function to find the most frequently occurring value in a dictionary.
Example: Input: `{'a': 3, 'b': 2, 'c': 3}`
Output: `3`

(d) Write a function that groups words by their first letter from a given list. The function should return a dictionary where the keys are the first letters and the values are lists of words.
Example: Input: `["apple", "banana", "apricot", "blueberry", "cherry"]`
Output: `{"a": ["apple", "apricot"], "b": ["banana", "blueberry"], "c": ["cherry"]}`

(e) Write a function that finds the key associated with the highest value in a dictionary.
Example: Input: `{'a': 10, 'b': 25, 'c': 17}`
Output: `'b'`

6. Sets:

(a) Write a function that returns the union, intersection, and difference of two sets.
Example:
Input: `{1,2,3}`, `{3,4,5}`
Output:
Union: `{1,2,3,4,5}`
Intersection: `{3}`
Difference (Set 1 - Set 2): `{1,2}`
Difference (Set 2 - Set 1): `{4,5}`

(b) Write a function that finds the symmetric difference between two sets.
Example:
Input: `{1,2,3}`, `{2,3,4}`
Output: `{1,4}`

(c) Write a function to check if two sets are disjoint.
Example:
Input: `{1,2,3}`, `{4,5,6}`
Output: `True`

Input: `{1,2,3}`, `{3,4,5}`
Output: `False`

Object-Oriented Programming (Chapter 3)

1. Define a class `Rectangle` in Python with:
   - Attributes: `length` and `width`.
   - A method `area()` that returns the area of the rectangle.
   - A method `perimeter()` that returns the perimeter of the rectangle.

2. Define a class `Circle` with an attribute `radius`. Include methods:
   - `area()` that calculates the area.
   - `circumference()` that calculates the circumference.

   Example: `Circle(5).area()` returns approximately 78.54.

3. Define a class `Student` with attributes `name`, `age`, and `grades`. Include methods:
   - `average()` returning the average of grades.
   - `is_passing()` returning `True` if the average grade is above a threshold (e.g., 60).

   Example: `Student("Alice",20,[80,90]).average()` returns 85.

4. Define a class `BankAccount` with attributes `balance` and methods `deposit()` and `withdraw()`.
   Example: After depositing 50 into an account initialized with 100, the balance is 150.

5. Implement inheritance by defining a superclass `Vehicle` with attributes `make` and `model`. Create subclasses `Car` and `Bike` with additional attributes `doors` for `Car` and `type` for `Bike`.
   Example: `Car("Ford","Mustang",4)` creates a car object.

Testing (Chapter 4)

1. Write a Python function `is_positive(n)` that returns `True` if `n` is positive and `False` otherwise. Use an `assert` statement to test your function.
   Example: Input: `5`, Output: `True`; Input: `-3`, Output: `False`

2. Write unit tests for the `Rectangle` class using Python's `unittest` framework.

3. Explain the concept of Test-Driven Development (TDD) and illustrate it by writing tests first for a simple function that calculates the factorial of a number.

4. Write tests that specifically check edge cases, incorrect usage, and error handling for a function that divides two numbers.

5. Explain why tests should be maintained even after they pass successfully.

Running Time Analysis (Chapter 5)

1. Implement a Python program that adds the first k natural numbers in two ways: using a loop and using the formula $S = \frac{k(k+1)}{2}$. Measure and compare the running time of both implementations.
   Example: Input: `k=1000`, Output: Time for loop: `X ms`, Time for formula: `Y ms`

2. Analyze the time complexity of common list operations such as appending, indexing, and slicing. Provide Python examples and justify the complexity of each operation.

3. Consider a dictionary with n elements. Measure the execution time of inserting, deleting, and accessing elements. Compare your results to the expected theoretical complexities.

4. Analyze the complexity of the exercises in the Basic Python section. Provide the worst-case time complexity for each function.

5. Explain the Big-O notation concept and determine the worst-case time complexity of the following function:

```python
def mystery_function(n):
    total = 0
    for i in range(n):
        for j in range(i):
            total += j
    return total
```