Python Coding Challenge Solutions

This file contains solutions to the 10 Python coding challenges presented in the README.md file.

1. String Reversal

```python
def reverse_string(s):
    return s[::-1]
```

2. List Filtering

```python
def filter_list(lst):
    return [x for x in lst if isinstance(x, int)]
```

3. Fibonacci Sequence

```python
def fibonacci(n):
    if n <= 1:
        return n
    a, b = 0, 1
    for _ in range(2, n + 1):
        a, b = b, a + b
    return b
```

4. Prime Number Check

```python
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
```

5. Dictionary Merge

```python
def merge_dicts(dict1, dict2):
    return {**dict1, **dict2}
```

6. Class Implementation

```python
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height
```

```
    def area(self):
        return self.width * self.height

    def perimeter(self):
        return 2 * (self.width + self.height)
```

7. File Parser

```
def file_stats(filename):
    with open(filename, 'r') as file:
        content = file.read()
        lines = content.split('\n')
        words = content.split()
        chars = len(content)
    return (len(lines), len(words), chars)
```

8. Data Aggregation

```
def aggregate_data(data):
    result = {}
    for d in data:
        for key, value in d.items():
            result[key] = result.get(key, 0) + value
    return result
```

9. Palindrome Check

```
import re

def is_palindrome(s):
    s = re.sub(r'[^a-zA-Z]', '', s.lower())
    return s == s[::-1]
```

10. List Sorting

```
def sort_by_character_count(strings):
    return sorted(strings, key=lambda x: (len(x), x))
```

Test Solutions

```
def test_solutions():
    # 1. String Reversal
    assert reverse_string("hello") == "olleh"
    assert reverse_string("Python") == "nohtyP"
    assert reverse_string("") == ""
    assert reverse_string("a") == "a"
```

```python
# 2. List Filtering
assert filter_list([1, 2, 'a', 'b']) == [1, 2]
assert filter_list([1, 'a', 'b', 0, 15]) == [1, 0, 15]
assert filter_list([1, 2, 3]) == [1, 2, 3]
assert filter_list(['a', 'b', 'c']) == []

# 3. Fibonacci Sequence
assert fibonacci(0) == 0
assert fibonacci(1) == 1
assert fibonacci(2) == 1
assert fibonacci(5) == 5
assert fibonacci(10) == 55

# 4. Prime Number Check
assert is_prime(2) == True
assert is_prime(3) == True
assert is_prime(4) == False
assert is_prime(29) == True
assert is_prime(100) == False

# 5. Dictionary Merge
assert merge_dicts({'a': 1, 'b': 2}, {'b': 3, 'c': 4}) == {'a': 1, 'b': 3, 'c': 4}
assert merge_dicts({'x': 1, 'y': 2}, {'y': 3, 'z': 4}) == {'x': 1, 'y': 3, 'z': 4}
assert merge_dicts({}, {'a': 1}) == {'a': 1}
assert merge_dicts({'a': 1}, {}) == {'a': 1}

# 6. Class Implementation
rect = Rectangle(5, 10)
assert rect.area() == 50
assert rect.perimeter() == 30
rect2 = Rectangle(3, 4)
assert rect2.area() == 12
assert rect2.perimeter() == 14

# 7. File Parser
# Note: This test assumes the existence of a 'sample.txt' file with specific content
# assert file_stats('sample.txt') == (3, 10, 51)

# 8. Data Aggregation
data = [
    {'a': 1, 'b': 2, 'c': 3},
    {'a': 4, 'b': 5, 'c': 6},
    {'a': 7, 'b': 8, 'c': 9}
]
assert aggregate_data(data) == {'a': 12, 'b': 15, 'c': 18}
```

```
    data2 = [
        {'x': 1, 'y': 2},
        {'x': 3, 'z': 4},
        {'y': 5, 'z': 6}
    ]
    assert aggregate_data(data2) == {'x': 4, 'y': 7, 'z': 10}

    # 9. Palindrome Check
    assert is_palindrome("A man, a plan, a canal: Panama") == True
    assert is_palindrome("race a car") == False
    assert is_palindrome("Was it a car or a cat I saw?") == True
    assert is_palindrome("hello world") == False

    # 10. List Sorting
    assert sort_by_character_count(["abc", "ab", "abcd", "a"]) == ["a", "ab", "abc", "abcd"]
    assert sort_by_character_count(["apple", "banana", "cherry", "date"]) == ["date", "apple
    assert sort_by_character_count(["zz", "abc", "aba", "abab", "z"]) == ["z", "zz", "aba",

    print("All tests passed successfully!")

if __name__ == "__main__":
    test_solutions()
```