

# Python Coding Challenges for Interview Preparation

This README contains 10 Python coding challenges ranging from easy to medium/hard level. Each challenge includes a description of the task and test cases to verify your solution.

## 1. String Reversal

**Task:** Write a function that reverses a given string.

**Test Cases:**

```
assert reverse_string("hello") == "olleh"
assert reverse_string("Python") == "nohtyP"
assert reverse_string("") == ""
assert reverse_string("a") == "a"
```

## 2. List Filtering

**Task:** Write a function that filters out all numbers from a list of mixed elements (strings and numbers).

**Test Cases:**

```
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

**Task:** Write a function that returns the nth number in the Fibonacci sequence.

**Test Cases:**

```
assert fibonacci(0) == 0
assert fibonacci(1) == 1
assert fibonacci(2) == 1
assert fibonacci(5) == 5
assert fibonacci(10) == 55
```

## 4. Prime Number Check

**Task:** Write a function that checks whether a given number is prime.

**Test Cases:**

```
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

**Task:** Write a function that merges two dictionaries. If there are duplicate keys, the value from the second dictionary should be used.

**Test Cases:**

```

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

**Task:** Implement a `Rectangle` class with methods to calculate area and perimeter.

**Test Cases:**

```

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

**Task:** Write a function that reads a text file and returns the number of lines, words, and characters in the file.

**Test Cases:**

```

# Assuming a file named 'sample.txt' with content:
# Hello, World!
# This is a test file.
# It has three lines.

assert file_stats('sample.txt') == (3, 10, 51)

```

## 8. Data Aggregation

**Task:** Write a function that takes a list of dictionaries and returns a new dictionary with the sum of values for each key.

**Test Cases:**

```

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

**Task:** Write a function that checks if a given string is a palindrome (reads the same forwards and backwards), ignoring spaces, punctuation, and letter casing.

**Test Cases:**

```

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

**Task:** Write a function that sorts a list of strings based on the length of each string. If two strings have the same length, sort them alphabetically.

**Test Cases:**

```

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

```

These challenges cover a wide range of Python basics and some intermediate concepts. Good luck with your interview preparation!