# Real-Life Recursion Problems in Python

## 1. Directory Traversal (List all files and folders)

```
import os
def list_files(path):
    for item in os.listdir(path):
        full_path = os.path.join(path, item)
        if os.path.isdir(full_path):
            list_files(full_path)
        else:
            print(full_path)
```

## 2. Calculating Compound Interest Recursively

```
def compound_interest(p, r, t):
    if t == 0:
        return p
    return compound_interest(p * (1 + r), r, t - 1)
```

## 3. Convert Decimal to Binary

```
def decimal_to_binary(n):
    if n == 0:
        return ""
    return decimal_to_binary(n // 2) + str(n % 2)
```

## 4. Count Number of Files in Nested Folders

```
import os
def count_files(path):
    count = 0
    for item in os.listdir(path):
        full_path = os.path.join(path, item)
        if os.path.isdir(full_path):
            count += count_files(full_path)
        else:
            count += 1
    return count
```

## 5. Calculate Total Salary with Annual Increment

```
def total_salary(salary, increment, years):
    if years == 0:
        return salary
    return salary + total_salary(salary * (1 + increment), increment, years - 1)
```

## 6. HTML/XML Tree Traversal

```
def traverse_dom_tree(node):
    print(node.tag)
    for child in node.children:
        traverse_dom_tree(child)
```

# Real-Life Recursion Problems in Python

## 7. Count Occurrences of Word in Nested Dictionary

```
def count_word(data, word):
    count = 0
    for key, value in data.items():
        if isinstance(value, dict):
            count += count_word(value, word)
        elif isinstance(value, str) and value == word:
            count += 1
    return count
```

## 8. Sum of Numbers in Nested List

```
def sum_nested_list(lst):
    total = 0
    for item in lst:
        if isinstance(item, list):
            total += sum_nested_list(item)
        else:
            total += item
    return total
```

## 9. Flatten Nested List

```
def flatten(lst):
    flat_list = []
    for item in lst:
        if isinstance(item, list):
            flat_list.extend(flatten(item))
        else:
            flat_list.append(item)
    return flat_list
```

## 10. Recursive Web Crawler (Simplified)

```
def web_crawler(url, depth):
    if depth == 0:
        return
    print(f'Crawling: {url}')
    # Pretend to find links (this would require requests + BeautifulSoup)
    links = ["http://example.com/a", "http://example.com/b"]
    for link in links:
        web_crawler(link, depth - 1)
```