Functions Problem Set

Table of contents

1 Problem 1: Even numbers

Write a Python function `filter_even_numbers` that takes a list of numbers and returns a new list containing only the even numbers.

- Example Input: `[1, 2, 3, 4]`
- Example Output: `[2, 4]`

- Solution

```
def filter_even_numbers(numbers):
  even_numbers = [] # create an empty list for the even numbers
  for num in numbers:
    if num % 2 == 0: # if the number is divisable by 2 then it is even
      even_numbers.append (num) # if so, add it to the list of even numbers
  return even_numbers

filter_even_numbers([1, 2, 3, 4])
```

```
[2, 4]
```

- Solution #2

```
def filter_even_numbers_v2(numbers):
  return [num for num in numbers if num % 2 == 0]

filter_even_numbers_v2([1, 2, 3, 4])
```

```
[2, 4]
```

2 Problem 2: Fizz Buzz

Write a Python function `fizzbuzz` that takes a number and returns `"Fizz"` if the number is divisible by 3, `"Buzz"` if the number is divisible by 5, and `"FizzBuzz"` if the number is divisible by both 3 and 5. Otherwise, returns the number.

- Solution

```
def fizzbuzz(number):
  if number % 3 == 0 and number % 5 == 0:
    return "FizzBuzz"
  elif number % 3 == 0:
    return "Fizz"
  elif number % 5 == 0:
    return "Buzz"
  else:
    return number
```

```
fizzbuzz(15)
```

```
'FizzBuzz'
```

```
fizzbuzz(5)
```

```
'Buzz'
```

```
fizzbuzz(3)
```

```
'Fizz'
```

```
fizzbuzz(1)
```

```
1
```

3 Problem 3: Second largest

Write a Python function `find_second_largest` that takes a list of positive numbers and returns the second largest number in the list *without* sorting the list.

- Example Input: `[2, 5, 7, 1, 8, 3, 9]`
- Example Output: `8`

- Solution

```
def find_second_largest(numbers):
  largest = -1
  second_largest = -1
  for num in numbers:
    if num > largest:
      second_largest = largest
      largest = num
    elif num > second_largest:
      second_largest = num
  return second_largest

find_second_largest([2, 5, 7, 1, 8, 3, 9])
```

```
8
```

4 Problem 4: Most common letter

Write a Python function `find_most_common_letter` that takes a string and returns the most common letter in the string.

- Example Input: `"Hello, World!"`
- Example Output: `'L'`

- Solution

```
def find_most_common_letter(string):
  letter_counts = {} # empty dictionary of the characters and their counts
  for letter in string.upper(): # for each character in the string (upper
   ↪  case)
    if letter in letter_counts: # if the char in the dictionary
      letter_counts[letter] += 1 # increase the count by 1
    else:
      letter_counts[letter] = 1 # otherwise, set the count to 1

  # Now let's find the largest count
  max_count = -1 # set the max count to -1
  max_letter = '' # set the letter with the max count to ''
  for letter in letter_counts:
    if letter_counts[letter] > max_count:
      max_count = letter_counts[letter]
      max_letter = letter

  return (max_letter)

find_most_common_letter("Hello, World!")
```

```
'L'
```

5 Problem 5: Recursive sum

Write a *recursive* Python function `recursive_sum` that takes a list of integers and returns the sum of all the numbers in the list.

- Example Input: `[1, 2, 3, 4, 5]`
- Example Output: `15`

- Solution

```
def recursive_sum(numbers):
  if not numbers: # if the list of numbers is empty
    return 0 # return zero
  else: # otherwise
    # return sum the first num in the list and
    # the sum of the remaining items in the list
    return numbers[0] + recursive_sum(numbers[1:])
recursive_sum([1, 2, 3, 4, 5])
```

```
15
```

6 Problem 6: Sum of Squares

Write a Python function `sum_of_squares` that takes a list of numbers and returns the sum of their squares.

- Example Input: `[1, 2, 3]`
- Example Output: `14`

- Solution

```
def sum_of_squares(numbers):
    return sum([num**2 for num in numbers])

sum_of_squares([1, 2, 3])
```

```
14
```

7 Problem 7: Unique words

Write a Python function `count_unique_words` that takes a list of strings and returns the number of unique words in the list.

- Example Input: `['apple', 'banana', 'apple', 'cherry', 'banana', 'date']`
- Example Output: `4`

- Solution

```
def count_unique_words(words):
  unique_words = [] # create an empty list to store the unique words
  for word in words: # for each word in the original list of words
    if word not in unique_words: # if the word is not the unique list
      unique_words.append(word) # add that word to the unique list
  return len(unique_words) # return the length of the list of the unique
   ↪ words

count_unique_words(['apple', 'banana', 'apple', 'cherry', 'banana', 'date'])
```

```
4
```

- Solution #2

```
def count_unique_words_v2(words):
  return len(set(words)) # return the length of the set version of the list
   ↪ words

count_unique_words_v2(['apple', 'banana', 'apple', 'cherry', 'banana',
 ↪ 'date'])
```

```
4
```

8 Problem 8: Higher-Order Functions

Write a function named `apply_operation` that takes a list of numbers, a function that performs an operation on a single number (e.g., square, cube), and applies that operation to each number in the list, returning a new list.

- Example Input: `apply_operation([1, 2, 3, 4], lambda x: x**2)`
- Example Output: `[1, 4, 9, 16]`

- Solution

```
def apply_operation(numbers, operation):
    return [operation(number) for number in numbers]


apply_operation([1, 2, 3, 4], lambda x: x**2)
```

```
[1, 4, 9, 16]
```

9 Problem 9: Flatten a Nested List

Write a *recursive* Python function `flatten` that takes a nested list (a list containing other lists) and returns a flat list containing all the elements in the nested list, in the same order.

- Example Input: `flatten([1, [2, 3], [4, [5, 6]], 7])`
- Example Output: `[1, 2, 3, 4, 5, 6, 7]`

- Solution

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


flatten([1, [2, 3], [4, [5, 6]], 7])
```

```
[1, 2, 3, 4, 5, 6, 7]
```

10 Problem 10: Reverse a List

Write a *recursive* function named `reverse_list` that takes a list and returns a new list with the elements in reverse order.

- Example Input: `[1, 2, 3, 4, 5]`
- Example Output: `[5, 4, 3, 2, 1]`

- Solution

```
def reverse_list(lst):
    if len(lst) == 0:
        return []
    else:
        return [lst[-1]] + reverse_list(lst[:-1])

reverse_list([1, 2, 3, 4, 5])
```

```
[5, 4, 3, 2, 1]
```

11 Problem 11: Find Missing Numbers

Given a list of unique integers sorted in increasing order, write a Python function named `find_missing` that returns a list of any missing integers in the sequence from the minimum to the maximum value.

- Example Input: `[1, 2, 4, 6, 7]`
- Example Output: `[3, 5]`

- Solution

```python
def find_missing(numbers):
    full_set = set(range(min(numbers), max(numbers) + 1))
    missing = full_set - set(numbers)
    return sorted(list(missing))

find_missing([1, 2, 4, 6, 7])
```

```
[3, 5]
```