ASSIGNMENT-2

1. Write a Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing', add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged.

Program:

```
# Define a function named add_string that takes one argument, 'str1'.
def add_string(str1):
    # Get the length of the input string 'str1' and store it in the variable
'length'.
    length = len(str1)

    # Check if the length of 'str1' is greater than 2 characters.
    if length > 2:
        # If the last three characters of 'str1' are 'ing', add 'ly' to the end.
        if str1[-3:] == 'ing':
            str1 += 'ly'
        else:
            # If the last three characters are not 'ing', add 'ing' to the end.
            str1 += 'ing'

    # Return the modified 'str1'.
    return str1

# Call the add_string function with different input strings and print

print(add_string('ab'))
print(add_string('abc'))
print(add_string('string'))
```

Output:

```
ab

abcing

stringly
```

2. Write a Python function that takes a list of words and return the longest word and the length of the longest one.

Program:

```
# Define a function named find_longest_word that takes a list of words as the
argument, 'words_list'.
def find_longest_word(words_list):
    # Create an empty list 'word_len' to store pairs of word lengths and the
```

```
corresponding words.
    word_len = []

    # Iterate through each word 'n' in the 'words_list'.
    for n in words_list:
        # Append a tuple containing the length of the word and the word itself to
'word_len'.
        word_len.append((len(n), n))

    # Sort the list 'word_len' based on the word lengths (ascending order).
    word_len.sort()

    # Return the length and the word of the last item in the sorted list (which is
the longest word).
    return word_len[-1][0], word_len[-1][1]

# Call the find_longest_word function with a list of words and store the result in
'result'.
result = find_longest_word(["PHP", "Exercises", "Backend"])

# Print the longest word and its length.
print("\nLongest word: ", result[1])
print("Length of the longest word: ", result[0])
```

Output:

```
Longest word:  Exercises
Length of the longest word:  9
```

3. Write a Python program to pack consecutive duplicates of a given list of elements into sublists.

Program:

```
# Import the 'groupby' function from the 'itertools' module
from itertools import groupby

# Define a function 'compress' that takes a list of numbers 'l_nums' as input
def compress(l_nums):
    # Use 'groupby' to group consecutive duplicates and return the unique keys
    return [key for key, group in groupby(l_nums)]

# Define a list 'n_list' with consecutive duplicate elements
n_list = [0, 0, 1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 8, 9, 4, 4]

# Print a message indicating the purpose of the following output
print("Original list:")

# Print the original list 'n_list'
print(n_list)
```

```
# Print a message indicating the purpose of the following output
print("\nAfter removing consecutive duplicates:")

# Call the 'compress' function with 'n_list' as an argument and print the result
with consecutive duplicates removed
print(compress(n_list))
```

Output:

```
Original list:
[0, 0, 1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 8, 9, 4, 4]

After removing consecutive duplicates:
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 4]
```

4. Write a Python program to find the item with the most occurrences in a given list.

Program:

```
# Define a function named 'most_frequent' that finds the most frequently occurring
item in a list.
def most_frequent(nums):
    return max(set(nums), key=nums.count)
    # Using 'set' to remove duplicates, then finding the item with the maximum
count using the 'max' function.

# Test the 'most_frequent' function with different lists.
print(most_frequent([1, 2, 1, 2, 3, 2, 1, 4, 2]))
# Find the most frequently occurring item in the list. (Expected output: 2)

nums = [2, 3, 8, 4, 7, 9, 8, 2, 6, 5, 1, 6, 1, 2, 3, 2, 4, 6, 9, 1, 2]
print("Original list:")
print(nums)
print("Item with the maximum frequency of the said list:")
print(most_frequent(nums))
# Find the most frequently occurring item in the list.

nums = [1, 2, 3, 1, 2, 3, 2, 1, 4, 3, 3]
print("\nOriginal list:")
print(nums)
print("Item with the maximum frequency of the said list:")
print(most_frequent(nums))
# Find the most frequently occurring item in the list.
```

Output:

```
2
Original list:
```

```
[2, 3, 8, 4, 7, 9, 8, 2, 6, 5, 1, 6, 1, 2, 3, 2, 4, 6, 9, 1, 2]
Item with maximum frequency of the said list:
2

Original list:
[1, 2, 3, 1, 2, 3, 2, 1, 4, 3, 3]
Item with maximum frequency of the said list:
3
```

5. Write a Python program to find the highest 3 values of corresponding keys in a dictionary.

Program:

```
# Import the 'nlargest' function from the 'heapq' module.
from heapq import nlargest

# Create a dictionary 'my_dict' with key-value pairs.
my_dict = {'a': 500, 'b': 5874, 'c': 560, 'd': 400, 'e': 5874, 'f': 20}

# Use the 'nlargest' function to find the three largest keys in the 'my_dict'
dictionary based on their values.
# The 'key' argument specifies that the values should be used for comparison.
three_largest = nlargest(3, my_dict, key=my_dict.get)

# Print the three largest keys found in the 'my_dict' dictionary.
print(three_largest)
```

Output:

```
['b', 'e', 'c']
```

6. Write a Python program to get the top three items in a shop.
Sample data: {'item1': 45.50, 'item2':35, 'item3': 41.30, 'item4':55, 'item5': 24}
Expected Output:
item4 55
item1 45.5
item3 41.3

Program:

```
# Import the 'nlargest' function from the 'heapq' module and the 'itemgetter'
function from the 'operator' module.
from heapq import nlargest
from operator import itemgetter

# Create a dictionary 'items' with keys representing items and values as their
corresponding prices.
items = {'item1': 45.50, 'item2': 35, 'item3': 41.30, 'item4': 55, 'item5': 24}
```

```
# Use the 'nlargest' function to find the three items with the largest prices from
the 'items' dictionary.
# The 'key' argument specifies that the second element (price) of each key-value
pair should be used for comparison.
for name, value in nlargest(3, items.items(), key=itemgetter(1)):
    # Print the name and price of the three items with the largest prices.
    print(name, value)
```

Output:

```
item4 55

item1 45.5

item3 41.3
```