Solutions of the Exercises of Module 1

1. Write a program that prints the numbers from 1 to 10 using a for loop.

In [ ]:
```
for num in range(1, 11):
    print(num)
```

1. Write a program that prints the numbers from 1 to 10 using a while loop.

In [ ]:
```
num = 1
while num <= 10:
    print(num)
    num += 1
```

1. Write a program that prints the even numbers from 1 to 10 using a for loop.

In [ ]:
```
for num in range(2, 11, 2):
    print(num)
```

1. Write a program that prints the odd numbers from 1 to 10 using a while loop.

In [ ]:
```
num = 1
while num <= 10:
    if num % 2 == 1:
        print(num)
    num += 1
```

1. Write a program that takes a number from the user and checks if it is positive, negative, or zero using an if-elif-else statement.

In [ ]:
```
num = int(input("Enter a number: "))
if num > 0:
    print("Positive")
elif num < 0:
    print("Negative")
else:
    print("Zero")
```

1. Write a program that takes two numbers from the user and prints the largest of the two using an if-else statement.

In [ ]:
```
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
if num1 > num2:
    print(num1, "is the largest")
else:
    print(num2, "is the largest")
```

1. Write a program that takes a list of numbers from the user and prints the sum of all the numbers using

a for loop.

In [ ]:
```
numbers = []
num = int(input("Enter a number (0 to stop): "))
while num != 0:
    numbers.append(num)
    num = int(input("Enter a number (0 to stop): "))

sum = 0
for num in numbers:
    sum += num
print("The sum of all the numbers is", sum)
```

1. Write a function to remove duplicates from a list while preserving the order.

In [2]:
```
def remove_duplicates(input_list):
    output_list = []
    seen = set()
    for item in input_list:
        if item not in seen:
            seen.add(item)
            output_list.append(item)
    return output_list
```

1. Write a function to find the second largest number in a list.

In [ ]:
```
def second_largest(numbers):
    # Get the unique values from the list
    unique_numbers = set(numbers)
    # Sort the list in descending order
    sorted_numbers = sorted(unique_numbers, reverse=True)
    # Return the second largest number
    return sorted_numbers[1]

# Example usage
numbers = [1, 2, 3, 4, 5, 5, 4, 3, 2, 1]
print(second_largest(numbers)) # Output: 4
```

1. Write a function to count the frequency of words in a string using a dictionary.

In [ ]:
```
def count_words(text):
    words = text.split()
    word_counts = {}
    for word in words:
        if word in word_counts:
            word_counts[word] += 1
        else:
            word_counts[word] = 1
    return word_counts

text = "the quick brown fox jumps over the lazy dog"
word_counts = count_words(text)
print(word_counts)
```

1. Write a function to find the common elements in two dictionaries.

In [ ]:
```
def common_elements(dict1, dict2):
    common = {}
```

```
    for key in dict1.keys():
        if key in dict2.keys():
            common[key] = dict1[key]
    return common

dict1 = {'a': 1, 'b': 2, 'c': 3}
dict2 = {'b': 2, 'c': 3, 'd': 4}
print(common_elements(dict1, dict2))
```

1. Write a function to find the top N items in a dictionary based on their values.

In [4]:
```
def top_n_items(d, n):
    sorted_dict = sorted(d.items(), key=sort_by_value, reverse=True)
    return [item[0] for item in sorted_dict[:n]]

def sort_by_value(item):
    return item[1]

# example usage
d = {"banana":10, "apple":20, "grapes":25}
top_n_items(d, 2)
```

Out[4]:
```
['grapes', 'apple']
```

This code defines two functions in Python. The first function, `top_n_items(d, n)`, takes a dictionary `d` and an integer `n` as input. The function first sorts the dictionary items based on their values using the sorted() function and a custom key function `sort_by_value()`. The `sorted()` function takes the dictionary `d.items()` as input, which returns a list of tuple pairs, where each tuple contains a key-value pair from the dictionary. The `key` argument of `sorted()` is set to `sort_by_value` to use the custom key function to sort the dictionary based on values. The `reverse` argument is set to `True` to sort the dictionary in descending order, so the items with the largest values appear first.

The sorted dictionary items are then sliced using the `[:n]` notation to return the top `n` items. Finally, a list comprehension is used to extract only the keys from the sorted items and return them as the final result of the function.

The second function, `sort_by_value(item)`, is the custom key function used in the `sorted()` function. This function takes a single argument `item`, which is a tuple containing a key-value pair from the dictionary. The function simply returns the value part of the tuple, so the `sorted()` function will sort the dictionary based on values.

In summary, the `top_n_items()` function finds the top `n` items in a dictionary based on their values, and returns a list of the keys of these items, sorted in descending order of their values.

In [ ]:
```

```