Functions – Part-II

#Immutable Data Types:

```
def modify_int(x):
    x += 10
    print("Inside the function:", x)

num = 5
modify_int(num)
print("Outside the function:", num)
```

```
def modify_string(s):
    s += " World!"
    print("Inside the function:", s)

greeting = "Hello"
modify_string(greeting)
print("Outside the function:", greeting)
```

```
def modify_tuple(t):
    # Tuples are immutable, so creating a new tuple
    t += (4, 5)
    print("Inside the function:", t)

coordinates = (1, 2, 3)
```

```
modify_tuple(coordinates)
print("Outside the function:", coordinates)




#Mutable Data Types:

def modify_list(lst):
    lst.append(4)
    lst[0] = 99
    print("Inside the function:", lst)

numbers = [1, 2, 3]
modify_list(numbers)
print("Outside the function:", numbers)




def modify_dict(d):
    d['new_key'] = 'new_value'
    print("Inside the function:", d)

my_dict = {'key1': 'value1', 'key2': 'value2'}
modify_dict(my_dict)
print("Outside the function:", my_dict)
```

```
def modify_set(s):
    s.add(4)
    print("Inside the function:", s)

my_set = {1, 2, 3}
modify_set(my_set)
print("Outside the function:", my_set)
```

```
# Assignment 1:
# Problem Statement:
# Write a function that takes a list of numbers
#as an argument and returns the sum of the squares
#of those numbers.

# Sample Input:

# numbers = [2, 3, 4]

# Sample Output:
# 29  # (2^2 + 3^2 + 4^2 = 4 + 9 + 16 = 29)

def sum_of_squares(nums):
```

```
    return sum(x**2 for x in nums)

numbers = [2, 3, 4]
result = sum_of_squares(numbers)
print(result)




# Assignment 2:
# Problem Statement:
# Create a function that accepts a dictionary and
#a key-value pair, and adds the pair to the dictionary.

# Sample Input:
# my_dict = {'a': 1, 'b': 2}
# key = 'c'
# value = 3

# Sample Output:
# {'a': 1, 'b': 2, 'c': 3}

def add_to_dict(d, key, value):
    d[key] = value
    return d

my_dict = {'a': 1, 'b': 2}
key = 'c'
value = 3
result = add_to_dict(my_dict, key, value)
```

```
print(result)
```

```
# Assignment 3:
# Problem Statement:
# Define a function that takes a string and returns a new string with all vowels converted to uppercase.

# Sample Input:
# text = "Hello, World!"

# Sample Output:
# "HellO, WOrld!"

def uppercase_vowels(input_string):
    vowels = "aeiouAEIOU"
    return ''.join(char.upper() if char in vowels else char for char in input_string)

text = "Hello, World!"
result = uppercase_vowels(text)
print(result)
```

Assignment 4:

Problem Statement:

Write a function that accepts a tuple and a value, and returns a new tuple with the value added at the end.

Sample Input:

my_tuple = (1, 2, 3)

value_to_add = 4

Sample Output:

(1, 2, 3, 4)

```
def add_to_tuple(t, value):
    return t + (value,)

my_tuple = (1, 2, 3)
value_to_add = 4
result = add_to_tuple(my_tuple, value_to_add)
print(result)
```

```
# Assignment 5:
# Problem Statement:
# Create a function that modifies a given list of names by removing any duplicates.

# Sample Input:
# names = ["Alice", "Bob", "Alice", "Charlie", "Bob"]

# Sample Output:
# ["Alice", "Bob", "Charlie"]

def remove_duplicates(name_list):
    return list(set(name_list))

names = ["Alice", "Bob", "Alice", "Charlie", "Bob"]
result = remove_duplicates(names)
print(result)
```

Scope of Variables:

1. Local Variable:

```
def example_function():
    # Local variable
    x = 10
    print("Inside function:", x)

example_function()
# Uncommenting the line below would result in an error since x is not defined outside the function.
```

```
# print("Outside function:", x)
```

2. Global Variable:

```
# Global variable
y = 20

def another_function():
    print("Inside function:", y)

another_function()
print("Outside function:", y)
```

3. Nonlocal Variable:

```
def outer_function():
    # Outer function variable
    z = 30

    def inner_function():
        nonlocal z
        # Nonlocal variable
        z += 5
        print("Inside inner function:", z)

    inner_function()
    print("Inside outer function:", z)

outer_function()
```

Anonymous Functions (Lambda Functions):

1. Simple Lambda Function:

```
# Lambda function to calculate the square of a number
square = lambda x: x**2
result = square(5)
print(result)
```

2. Lambda Function in a Higher-Order Function:

```
# Using lambda function in a higher-order function (map)
numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(lambda x: x**2, numbers))
print(squared_numbers)
```

3. Lambda Function in Filter:

```
# Using lambda function in filter to get even numbers
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)
```

Assignments:

1. **Scope Assignment:**

    - Write a function that takes a parameter and defines a local variable inside the function. Try to access that local variable outside the function and observe the result.

    Solution –

    ```
    def local_variable_function(parameter):
        # Local variable
        x = parameter
        print("Inside function:", x)

    local_variable_function(42)

    # Uncommenting the line below would result in an error since x is not defined outside the function.
    # print("Outside function:", x)
    ```

Global Variable Assignment:

- Create a global variable and a function that modifies the global variable inside the function. Print the global variable before and after calling the function.

Solution –

```
# Global variable
global_var = 100

def modify_global_variable():
    global global_var
    global_var += 50
    print("Inside function:", global_var)

print("Before function:", global_var)
modify_global_variable()
```

```
print("After function:", global_var)
```

Nonlocal Variable Assignment:

- Write a nested function where the outer function defines a variable, and the inner function modifies that variable using the **nonlocal** keyword. Print the variable inside and outside the functions.

  Solution –

```
def outer_function():
    # Outer function variable
    z = 30

    def inner_function():
        nonlocal z
        # Nonlocal variable
        z += 5
        print("Inside inner function:", z)

    inner_function()
    print("Inside outer function:", z)

outer_function()
```

Lambda Function Assignment:

- Write a lambda function that calculates the cube of a number. Use this lambda function in a list comprehension to find the cubes of numbers from 1 to 5.

  Solution –

```
# Lambda function to calculate the cube of a number
cube = lambda x: x**3

# Using the lambda function in a list comprehension
```

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

cubes = [cube(num) for num in numbers]

print(cubes)
```

Higher-Order Function Assignment:

- Create a list of strings and use a lambda function in the **sorted** function to sort the strings based on their lengths.

  Solution –

```
# List of strings

string_list = ["apple", "banana", "kiwi", "orange", "grape"]


# Sorting strings based on their lengths using a lambda function in sorted()

sorted_strings = sorted(string_list, key=lambda s: len(s))

print(sorted_strings)
```