Task-1

Understanding list in Python

> a list is a collection data type that is ordered and mutable, meaning that the elements can be changed after the list is created.

```
# Creating a list
my_list = [1, 2, 3, 'apple', 'banana', 4.5]

# Printing the list
print("Original List:", my_list)

# Accessing elements
print("First element:", my_list[0])
print("Last element:", my_list[-1])

# Modifying elements
my_list[2] = 'orange'
print("Modified List:", my_list)

# Adding elements
my_list.append('grape')
print("List after appending:", my_list)

# Removing elements
my_list.remove('banana')
print("List after removing 'banana':", my_list)

# Sorting the list
num_list = [5, 3, 1, 4, 2]
num_list.sort()
print("Sorted num_list:", num_list)
```

```
Original List: [1, 2, 3, 'apple', 'banana', 4.5]
First element: 1
Last element: 4.5
Modified List: [1, 2, 'orange', 'apple', 'banana', 4.5]
List after appending: [1, 2, 'orange', 'apple', 'banana', 4.5, 'grape']
List after removing 'banana': [1, 2, 'orange', 'apple', 4.5, 'grape']
Sorted num_list: [1, 2, 3, 4, 5]
```

Dictionaries

A dictionary is a collection of key-value pairs. It is unordered, mutable, and indexed by keys, which are unique and must be immutable types (such as strings, numbers, or tuples with immutable elements).

```
# Creating a dictionary
my_dict = {
    'name': 'Jitshirwa',
    'age': 21,
    'city': 'Delhi',
    'is_student': True,
    'grades': [85, 90, 78]
}

# Printing the dictionary
print("Original Dictionary:", my_dict)

# Accessing values
print("Name:", my_dict['name'])
print("Age:", my_dict['age'])

# Modifying values
my_dict['age'] = 24
print("Updated Dictionary:", my_dict)

# Adding a new key-value pair
my_dict['email'] = 'j@example.com'
print("Dictionary after adding email:", my_dict)

# Removing a key-value pair
del my_dict['city']
print("Dictionary after removing city:", my_dict)

# Iterating over keys and values
for key, value in my_dict.items():
    print(f"{key}: {value}")

# Checking if a key exists
if 'name' in my_dict:
    print("Name exists in the dictionary")

# Using the get method to access a value
email = my_dict.get('email', 'Email not found')
print("Email:", email)

# Using the keys and values methods
print("Keys:", list(my_dict.keys()))
print("Values:", list(my_dict.values()))
```

```
Original Dictionary: {'name': 'Jitshirwa', 'age': 21, 'city': 'Delhi', 'is_stude
Name: Jitshirwa
Age: 21
Updated Dictionary: {'name': 'Jitshirwa', 'age': 24, 'city': 'Delhi', 'is_studer
Dictionary after adding email: {'name': 'Jitshirwa', 'age': 24, 'city': 'Delhi',
Dictionary after removing city: {'name': 'Jitshirwa', 'age': 24, 'is_student': T
name: Jitshirwa
age: 24
is_student: True
grades: [85, 90, 78]
email: j@example.com
Name exists in the dictionary
Email: j@example.com
Keys: ['name', 'age', 'is_student', 'grades', 'email']
```

```
    Values: ['Jitshirwa', 24, True, [85, 90, 78], 'j@example.com']
```

SETS

A set is an unordered collection of unique elements. Sets are mutable, meaning you can add or remove elements after a set is created.

```
# Creating a set
my_set = {1, 2, 3, 'apple', 'banana', 4.5}
print("Original Set:", my_set)

# Creating a set using the set() function
another_set = set([1, 2, 2, 3, 4, 4, 5])
print("Another Set:", another_set)

# Adding elements to a set
my_set.add('orange')
print("Set after adding 'orange':", my_set)

# Removing elements from a set
my_set.remove('apple')
print("Set after removing 'apple':", my_set)


my_set.discard('pear')
print("Set after discarding 'pear':", my_set)

# Checking if an element is in the set
print("Is 'banana' in the set?", 'banana' in my_set)

# Iterating over a set
for item in my_set:
    print(item)

# Set operations (union, intersection, difference, symmetric difference)
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}

# Union
union_set = set1.union(set2)
print("Union of set1 and set2:", union_set)

# Intersection
intersection_set = set1.intersection(set2)
print("Intersection of set1 and set2:", intersection_set)

# Difference
difference_set = set1.difference(set2)
print("Difference of set1 and set2:", difference_set)

# Symmetric Difference
sym_diff_set = set1.symmetric_difference(set2)
print("Symmetric Difference of set1 and set2:", sym_diff_set)
```

```
Original Set: {1, 2, 3, 4.5, 'apple', 'banana'}
Another Set: {1, 2, 3, 4, 5}
Set after adding 'orange': {1, 2, 3, 4.5, 'apple', 'orange', 'banana'}
Set after removing 'apple': {1, 2, 3, 4.5, 'orange', 'banana'}
Set after discarding 'pear': {1, 2, 3, 4.5, 'orange', 'banana'}
Is 'banana' in the set? True
1
2
3
4.5
orange
banana
Union of set1 and set2: {1, 2, 3, 4, 5, 6}
Intersection of set1 and set2: {3, 4}
Difference of set1 and set2: {1, 2}
Symmetric Difference of set1 and set2: {1, 2, 5, 6}
```