

## Lab Experiment 7: Python

### Question 1: Tuple Basics

1. Create a tuple `my_tuple` with elements (10, 20, 30, 40, 50).
2. Print the length of `my_tuple`.
3. Print the element at index 2.
4. Print the last element of the tuple.

### Question 2: Tuple Manipulation

Write a Python program that does the following:

1. Create a tuple `original_tuple` with elements (10, 20, 30, 40, 50).
2. Print the length of `original_tuple`.
3. Print the element at index 3.
4. Print the last element of the tuple.
5. Convert `original_tuple` to a list `new_list`.
6. Add a new element 60 to `new_list`.
7. Convert `new_list` back to a tuple `new_tuple`.
8. Print `new_tuple`.

### Question 3: Unpacking Tuples

Write a program that unpacks a tuple into multiple variables. The tuple is (10, 20, 30) and the variables should be `a`, `b`, and `c`. Print the values of `a`, `b`, and `c`.

### Question 4: Tuple Concatenation

Write a Python program that concatenates two tuples:

1. Create two tuples, `tuple1` with elements (1, 2, 3) and `tuple2` with elements (4, 5, 6).
2. Concatenate `tuple1` and `tuple2` into a new tuple `result_tuple`.
3. Print `result_tuple`.

### Question 5: Tuple Concatenation

1. Create two tuples, `tuple1` with elements (1, 2, 3) and `tuple2` with elements (4, 5, 6).
2. Concatenate `tuple1` and `tuple2` into a new tuple `result_tuple`.
3. Print `result_tuple`.

### Question 6: Tuple Packing and Unpacking

Write a Python program that demonstrates tuple packing and unpacking:

1. Create variables `name`, `age`, and `country`.
2. Pack these variables into a tuple called `person`.
3. Unpack the `person` tuple into `name`, `age`, and `country` variables.
4. Print these variables.

### Question 7: Tuple Unpacking

Write a program that unpacks a tuple into variables `a`, `b`, and `c`. The tuple is (10, 20, 30).

### Question 8: Tuple Comprehension

Write a Python program that creates a tuple of squares of numbers:

1. Use tuple comprehension to create a tuple `squares_tuple` that contains squares of numbers from 1 to 10.
2. Print `squares_tuple`.

#### **Question 9: Frequency Count in Tuple**

Write a Python program to count the frequency of elements in a tuple:

1. Create a tuple `test_tuple` with repeated elements (1, 2, 3, 4, 1, 2, 1, 4, 5).
2. Count the frequency of each element and store it in a dictionary `frequency_dict`.
3. Print `frequency_dict`.

#### **Question 10: Immutable Sets in Tuple**

Write a Python program that creates a tuple of immutable sets:

1. Create a tuple `immutable_sets` containing two sets: {1, 2, 3} and {3, 4, 5}.
2. Print `immutable_sets`.

#### **Question 11: Convert List to Tuple**

1. Create a list `my_list` with elements [10, 20, 30].
2. Convert `my_list` to a tuple `my_tuple`.
3. Print `my_tuple`.

#### **Question 12: Reverse a Tuple**

1. Create a tuple `numbers` with elements (1, 2, 3, 4, 5).
2. Write a program to reverse the elements of `numbers`.
3. Print the reversed tuple.

#### **Question 13: Find Maximum Element**

1. Create a tuple `num_tuple` with elements (10, 20, 30, 40, 50).
2. Write a program to find the maximum element in `num_tuple`.
3. Print the maximum element.

#### **Question 14: Tuple Membership Check**

Write a Python program to check if an element exists in a tuple:

1. Create a tuple `check_tuple` with elements (10, 20, 30, 40, 50).
2. Prompt the user to enter a number.
3. Check if the entered number exists in `check_tuple` and print an appropriate message.

#### **Question 15: Check Tuple Membership**

1. Create a tuple `test_tuple` with elements (1, 2, 3, 4, 5).
2. Prompt the user to enter a number.
3. Check if the entered number exists in `test_tuple` and print an appropriate message.

#### **Question 16: Zip and Unzip Tuples**

Write a Python program that demonstrates zip and tuple unpacking:

1. Create two tuples `names` and `ages` containing names and corresponding ages.
2. Use `zip` to combine `names` and `ages` into a `person_info` tuple.
3. Unpack `person_info` into `name` and `age` variables.
4. Print `name` and `age`.

**Question 17: Tuple Sorting**

Write a Python program that sorts a tuple of tuples based on the second element of each tuple:

1. Create a tuple of tuples students with elements (("Alice", 22), ("Bob", 19), ("Charlie", 25)).
2. Sort students based on the second element of each tuple.
3. Print the sorted students.

**Question 18: Tuple Comprehension**

1. Use tuple comprehension to create a tuple squares\_tuple that contains squares of numbers from 1 to 10.
2. Print squares\_tuple.

**Question 19: Count Frequency**

1. Create a tuple test\_tuple with elements (1, 2, 3, 4, 1, 2, 1, 4, 5).
2. Count the frequency of each element and store it in a dictionary frequency\_dict.
3. Print frequency\_dict.

**Question 20: Tuple Slice**

1. Create a tuple my\_tuple with elements (1, 2, 3, 4, 5, 6, 7, 8, 9, 10).
2. Slice my\_tuple to create a new tuple slice\_tuple with elements from index 3 to index 7.
3. Print slice\_tuple.

**Question 21: Check Tuple Equality**

1. Create two tuples tuple1 and tuple2 with the same elements.
2. Write a program to check if tuple1 and tuple2 are equal.
3. Print "Equal" or "Not Equal" based on the comparison.

**Question 22: Tuple Concatenation and Sorting**

1. Create two tuples tuple1 with elements (5, 8, 2) and tuple2 with elements (3, 6, 1).
2. Concatenate tuple1 and tuple2 into a new tuple result\_tuple.
3. Sort result\_tuple in descending order.
4. Print the sorted result\_tuple.

**Question 23: Unique Elements in Tuple**

1. Create a tuple mixed\_tuple with elements (1, 2, 3, 4, 5, 1, 2, 3).
2. Write a program to create a new tuple unique\_tuple with unique elements from mixed\_tuple.
3. Print unique\_tuple.

**Question 24: Tuple Intersection**

1. Create two tuples tuple1 with elements (1, 2, 3, 4) and tuple2 with elements (3, 4, 5, 6).
2. Write a program to find the intersection of tuple1 and tuple2.
3. Print the intersection.

**Question 25: Tuple Flattening**

1. Create a tuple nested\_tuple with nested tuples ((1, 2), (3, 4), (5, 6)).

2. Write a program to flatten nested\_tuple into a single-level tuple.
3. Print the flattened tuple.

**Question 26: Tuple Range Check**

1. Create a tuple range\_tuple with elements (10, 20, 30, 40, 50).
2. Prompt the user to enter a number.
3. Check if the entered number falls within the range of elements in range\_tuple.
4. Print an appropriate message.