Q1. my\_tuple = ( (1, 2, 3), [4, 5, 6], ('apple', 'banana', 'cherry'), [7, 8, 9] )

* Retrieve the last element of the second list in my\_tuple
* Extract and print the first two elements from the list at index 3
* Modify an element inside the list at index 1 (change 5 to 10)
* Print all elements from my\_tuple on a new line (using for loop)
* Create a new tuple by concatenating the first and last elements of my\_tuple

**# 1. Last element of the second list in my\_tuple**

**last\_element\_second\_list = my\_tuple[1][-1]**

**print(last\_element\_second\_list)**

**# 2. First two elements from the list at index 3**

**first\_two\_elements\_at\_index\_3 = my\_tuple[3][:2]**

**print(first\_two\_elements\_at\_index\_3)**

**# 3. Modify element inside list at index 1 (change 5 to 10)**

**my\_tuple[1][1] = 10**

**print(my\_tuple[1])**

**# 4. Print all elements from my\_tuple on a new line**

**for element in my\_tuple:**

**print(element)**

**# 5. Create a new tuple by concatenating the first and last elements of my\_tuple**

**new\_tuple = my\_tuple[0] + tuple(my\_tuple[3])**

**print(new\_tuple)**

Q2. inventory = { "The Great Gatsby": {"price": 10.99, "quantity": 5}, "Rich dad poor dad": {"price": 8.99, "quantity": 8}, "Harry Potter": {"price": 12.50, "quantity": 2}, }

* Add a new book in inventory using method
* Get all book titles using method
* Get the total value of the inventory using method and for loop
* Remove a book using method if it's out of stock

**# Adding a new book using the update() method**

**inventory.update({ "think and grow rich": {"price": 9.99, "quantity": 3} })**

**print(inventory)**

**# Get all book titles using the keys() method**

**book\_titles = list(inventory.keys())**

**print(book\_titles)**

**#Get total value of the inventory using method and for loop**

**total\_value = 0**

**for book in inventory.values():**

**total\_value += book["price"] \* book["quantity"]**

**print("Total inventory value:", total\_value)**

**# Check and remove the book if out of stock (quantity 0)**

**if inventory["Harry Potter"]["quantity"] == 0:**

**del inventory["Harry Potter"]**

**print("Updated inventory:", inventory)**

Q3. An e-commerce store stores information about its products in a nested dictionary. The outer dictionary uses product IDs as keys, and the inner dictionary stores product details like name, category, price, and stock quantity.

products = { 101: {"name": "Laptop", "category": "Electronics", "price": 1200, "stock": 50}, 102: {"name": "Shirt", "category": "Apparel", "price": 25, "stock": 200}, 103: {"name": "Coffee Maker", "category": "Home Appliances", "price": 80, "stock": 30} }

* Update the price of the "Laptop" product to from 1200 to 1400
* Increase the stock of the "Shirt" product (add 50 more units)
* Add a new product (e.g., "Smartphone")
* List all product names and their categories (using loop)
* Print the details of a specific product (e.g., Coffee Maker)

**# 1. Update the price of the "Laptop" product to 1400**

**products[101]["price"] = 1400**

**# 2. Increase the stock of the "Shirt" product (add 50 more units)**

**products[102]["stock"] += 50**

**# 3. Add a new product (e.g., "Smartphone")**

**products[104] = {"name": "Smartphone", "category": "Electronics", "price": 800, "stock": 100}**

**#4. List all product names and their categorie**

**print("Product Names and Categories:")**

**for product in products.values():**

**print(f"Product Name: {product['name']}, Category: {product['category']}")**

**# 5. Print the details of the "Coffee Maker"**

**product product\_id = 103**

**print("\nDetails of Product ID 103 (Coffee Maker):")**

**print(products[product\_id])**

Q4. You are given a list that contains some duplicate items. Remove the duplicates by converting the list to a set.

**Task:**

* Convert the given list to a set to remove duplicates.
* Convert the set back to a list to show the result.
* Print both the original list and the list without duplicates

shopping\_list = ["apple", "banana", "apple", "orange", "banana", "grape"]

**unique\_items = set(shopping\_list)**

**unique\_shopping\_list = list(unique\_items)**

**print("Original shopping list:", shopping\_list)**

**print("Shopping list without duplicates:", unique\_shopping\_list)**

Q5. You are managing a list of students enrolled in two different courses. You need to perform various set operations to understand the student enrollment. Task: Create two sets: one for students in "Course A" and one for students in "Course B". Find the students who are enrolled in both courses (intersection). Find the students who are enrolled in at least one of the courses (union). Find the students who are in "Course A" but not in "Course B" (difference). Find students who are only in one of the two courses (symmetric difference). course\_a = {"John", "Alice", "Bob", "David"} course\_b = {"Alice", "Eve", "Charlie", "David"}

* Find students who are enrolled in both courses (intersection)
* Find students who are enrolled in at least one of the courses (union)
* Find students who are in Course A but not in Course B (difference)
* Find students who are only in one of the two courses (symmetric difference)

**# 1. Find students who are enrolled in both courses (intersection)**

**intersection\_students = course\_a & course\_b**

**print("Students enrolled in both courses:", intersection\_students)**

**# 2. Find students who are enrolled in at least one of the courses (union)**

**union\_students = course\_a | course\_b**

**print("Students enrolled in at least one of the courses:", union\_students)**

**# 3. Find students who are in Course A but not in Course B (difference)**

**difference\_students = course\_a - course\_b**

**print("Students in Course A but not in Course B:", difference\_students)**

**# 4. Find students who are only in one of the two courses (symmetric difference) symmetric\_difference\_students = course\_a ^ course\_b**

**print("Students only in one of the two courses:", symmetric\_difference\_students)**

Q6. Write a Python program that calculates the sum of all even numbers between 1 and 50 (inclusive) using a for loop.

1. Initialize a variable to hold the sum (e.g., sum\_of\_evens).
2. Use a for loop to iterate through all numbers from 1 to 50.
3. Inside the loop, check if the number is even.
4. If the number is even, add it to sum\_of\_evens.
5. After the loop finishes, print the total sum of all even numbers between 1 and 50.

**sum\_of\_evens = 0**

**for number in range(1, 51):**

**if number % 2 == 0:**

**sum\_of\_evens += number**

**print("The sum of even numbers between 1 and 50 is:", sum\_of\_evens)**

Q7. Write a Python program that prints the multiplication table for a given number (between 1 and 10) using a for loop.

1. Ask the user for a number between 1 and 10 .
2. Use a for loop to iterate through the numbers from 1 to 10.
3. For each iteration, calculate the product of the given number and the current number in the loop.
4. Print the result in a readable format (e.g., 5 x 1 = 5).
5. Ensure the program works correctly for any number between 1 and 10.

**number = int(input("Enter a number between 1 and 10: ")**

**if 1 <= number <= 10:**

**print(f"Multiplication Table for {number}:")**

**for i in range(1, 11):**

**print(f"{number} x {i} = {number \* i}")**

**else:**

**print("Please enter a number between 1 and 10.")**

Q8. **Power of a Number**

Write a Python program that takes a number and prints the powers of the number (starting from 1 to 10) using a while loop. For example, if the user inputs 3, the output should be:  
3^1 = 3  
3^2 = 9  
3^3 = 27  
... up to 3^10.

**number = int(input("Enter a number: "))**

**i = 1**

**while i <= 10:**

**power = number \*\* i**

**print(f"{number}^{i} = {power}")**

**i += 1**

Q9. **Problem Statement:** Write a Python program that takes an integer input from the user and counts down from that number to 0. The program should display the current number at each step until it reaches 0, at which point it should print a message indicating the countdown is finished.

**n = int(input("Enter a number to start the countdown: "))**

**while n >= 0:**

**print(n)**

**n -= 1**

**print("Countdown finished!")**

Q10. Write a Python program that takes an integer input n from the user and generates a right-angled triangle pattern of stars (\*) with n rows. The first row should contain 1 star, the second row 2 stars, the third row 3 stars, and so on until the nth row.

Ensure the stars in each row are separated by spaces, and each row starts on a new line.

Pattern:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

**n = int(input("Enter the number of rows: "))**

**for i in range(1, n + 1):**

**for j in range(i):**

**print("\*", end=" ")**

**print()**