Program 1:- Write a program to create one array from another array.

```
#Write a program to create one array from another array
# Original array
#original_array = [1, 2, 3, 4, 5]
input_array =input("Enter elements of the array separated by spaces: ").split()

# Convert the input array elements to integers
input_array = [int(x) for x in input_array]
# Using the copy method
copied_array = input_array.copy()

# Using list slicing
copied_array_slicing = input_array[:]

print("Original array:", input_array)
print("Copied array using copy method:", copied_array)
print("Copied array using slicing:", copied_array_slicing)
```

Output:-

```
Enter elements of the array separated by spaces: 12 45 23 67 89
Original array: [12, 45, 23, 67, 89]
Copied array using copy method: [12, 45, 23, 67, 89]
Copied array using slicing: [12, 45, 23, 67, 89]
```

Program 2:-Create a program to retrieve, display and update only a range of elements from an array using indexing and slicing in arrays.

```
#import array
original_array =input("Enter elements of the array separated by spaces: ").split()
print("current array : ",original_array)
start_index = int(input("Enter the start index for the subarray: "))
end_index = int(input("Enter the end index for the subarray: "))
subarray = original_array[start_index : end_index]
print("Retrieved subarray:", subarray)
update = input("Do you want to update this subarray? (y/n): ").strip().lower()
if update == 'y':
    new_values = input("Enter the new values for the subarray, separated by spaces: ").split()
    new_values = [int(x) for x in new_values]
    original_array[start_index : end_index] = new_values
    print(original_array)
else:
    print("No updates made to the array.")
```

Output:-

```
Enter elements of the array separated by spaces: 12 56 34 29 78 99
current array :  ['12', '56', '34', '29', '78', '99']
Enter the start index for the subarray: 2
Enter the end index for the subarray: 6
Retrieved subarray: ['34', '29', '78', '99']
Do you want to update this subarray? (y/n): y
Enter the new values for the subarray, separated by spaces: 26 55 66 77
['12', '56', 26, 55, 66, 77]
```

Program 3:-Write a program to understand various methods of array class mentioned: append, insert, remove, pop, index, tolist and count.

```
import array
def display_menu():
    print("\nArray Operations Menu:")
    print("1. Append an element")
    print("2. Insert an element")
    print("3. Remove an element")
    print("4. Pop an element")
    print("5. Find the index of an element")
    print("6. Convert array to list")
    print("7. Count occurrences of an element")
    print("8. Display array")
    print("9. Exit")

def main():
    # Initialize an array of type 'i' (integer) with elements entered by the user
    input_array = list(map(int, input("Enter elements of the array separated by spaces: ").split()))
    arr = array.array('i', input_array)

    while True:
        display_menu()
        choice = input("Enter your choice (1-9): ")

        if choice == '1':
            element = int(input("Enter the element to append: "))
            arr.append(element)
            print(f"Element {element} appended.")

        elif choice == '2':
            element = int(input("Enter the element to insert: "))
            position = int(input("Enter the position to insert the element at: "))
            arr.insert(position, element)
            print(f"Element {element} inserted at position {position}.")

        elif choice == '3':
            element = int(input("Enter the element to remove: "))
            try:
                arr.remove(element)
                print(f"Element {element} removed.")
            except ValueError:
                print(f"Element {element} not found in array.")

        elif choice == '4':
            try:
                element = arr.pop()
                print(f"Element {element} popped from array.")
            except IndexError:
                print("Array is empty. Cannot pop an element.")

        elif choice == '5':
            element = int(input("Enter the element to find the index of: "))
            try:
```

```
            index = arr.index(element)
            print(f"Element {element} found at index {index}.")
        except ValueError:
            print(f"Element {element} not found in array.")

    elif choice == '6':
        lst = arr.tolist()
        print("Array converted to list:", lst)

    elif choice == '7':
        element = int(input("Enter the element to count occurrences of: "))
        count = arr.count(element)
        print(f"Element {element} occurs {count} times in array.")

    elif choice == '8':
        print("Current array:", arr)

    elif choice == '9':
        print("Exiting the program.")
        break

    else:
        print("Invalid choice. Please enter a number between 1 and 9.")

if __name__ == "__main__":
    main()
```

Output:-

```
Enter elements of the array separated by spaces: 12 56 34 29 78 99

Array Operations Menu:
1. Append an element
2. Insert an element
3. Remove an element
4. Pop an element
5. Find the index of an element
6. Convert array to list
7. Count occurrences of an element
8. Display array
9. Exit
Enter your choice (1-9): 1
Enter the element to append: 10
Element 10 appended.
```

```
Enter your choice (1-9): 2
Enter the element to insert: 22
Enter the position to insert the element at: 2
Element 22 inserted at position 2.
```

```
Enter your choice (1-9): 3
Enter the element to remove: 5
Element 5 not found in array.
```

```
Enter your choice (1-9): 4
Element 10 popped from array.
```

Program 4:-Write a program to sort the array elements using bubble sort technique.

```
import array

def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
```

```
            arr[j], arr[j + 1] = arr[j + 1], arr[j]

def main():
    # Initialize an array of type 'i' (integer) with elements entered by the user
    input_array = list(map(int, input("Enter elements of the array separated by spaces: ").split()))
    arr = array.array('i', input_array)

    print("Original array:", arr)

    bubble_sort(arr)

    print("Sorted array:", arr)

if __name__ == "__main__":
    main()
```

Output:-

```
Enter elements of the array separated by spaces: 12 45 33 89 67 23
Original array: array('i', [12, 45, 33, 89, 67, 23])
Sorted array: array('i', [12, 23, 33, 45, 67, 89])
```

Program 5:-Create a program to search the position of an element in an array using index() method of array class.

```
import array
# Initialize an array of type 'i' (integer) with elements entered by the user
input_array = list(map(int, input("Enter elements of the array separated by spaces: ").split()))
arr = array.array('i', input_array)
print("Array:", arr)
    # Get the element to search for
element = int(input("Enter the element to find the index of: "))
try:
    index = arr.index(element)
    print(f"Element {element} found at index {index}.")
except ValueError:
    print(f"Element {element} not found in array.")
```

Output:-

```
Enter elements of the array separated by spaces: 12 56 34 89 70 32 99
Array: array('i', [12, 56, 34, 89, 70, 32, 99])
Enter the element to find the index of: 70
Element 70 found at index 4.
```

Program 6:-Write a program to generate prime numbers with the help of a function to test prime or not.

```
def is_prime(n):
    """Check if a number is prime."""
    if n <= 1:
            return False
    if n <= 3:
           return True
    if n % 2 == 0 or n % 3 == 0:
       return False
    i = 5
```

```
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

def generate_primes(limit):
    """Generate all prime numbers up to a given limit."""
    primes = []
    for num in range(2, limit + 1):
        if is_prime(num):
            primes.append(num)
    return primes

# Take user input for the limit
limit = int(input("Enter the limit up to which you want to generate prime numbers: "))

# Generate and print prime numbers
prime_numbers = generate_primes(limit)
print(f"Prime numbers up to {limit}: {prime_numbers}")
```

Output:-

```
Enter the limit up to which you want to generate prime numbers: 7
Prime numbers up to 7: [2, 3, 5, 7]
```

Program 7:-Write a python program that removes any repeated items from a list so that each item appears at most once. For instance, the list [1,1,2,3,4,3,0,0] would become [1,2,3,4,0].

```
def remove_duplicates(input_list):
    """Remove duplicates from the list while preserving the order."""
    seen = set()
    result = []
    for item in input_list:
        if item not in seen:
            seen.add(item)
            result.append(item)
    return result
# Take user input for the list
input_str = input("Enter the list of items separated by commas: ")
input_list = [int(item.strip()) for item in input_str.split(',')]
# Remove duplicates and print the result
unique_list = remove_duplicates(input_list)
print(f"List after removing duplicates: {unique_list}")
```

Output:-

```
Enter the list of items separated by commas: 12,56,34,12,34,78,89,59
List after removing duplicates: [12, 56, 34, 78, 89, 59]
```

Program 8:-Write a program to pass a list to a function and display it.

```
def display_list(input_list):
    """Display the list."""
    print(f"The list is: {input_list}")

# Take user input for the list
```

```
input_str = input("Enter the list of items separated by commas: ")
input_list = [item.strip() for item in input_str.split(',')]

# Pass the list to the function and display it
display_list(input_list)
```

Output:-

```
Enter the list of items separated by commas: 12,45,32,67,89,44,77,20
The list is: ['12', '45', '32', '67', '89', '44', '77', '20']
```

Program 9:-Write a program to demonstrate the use of Positional argument, keyword argument and default arguments.

```
def show_details(name, age, country="Unknown", occupation="Student"):
    print(f"Name: {name}")
    print(f"Age: {age}")
    print(f"Country: {country}")
    print(f"Occupation: {occupation}")

# Taking input from the user
name = input("Enter your name: ")
age = input("Enter your age: ")

# Demonstrating positional arguments
print("\nUsing positional arguments:")
show_details(name, age)

# Demonstrating keyword arguments
print("\nUsing keyword arguments:")
show_details(name=name, age=age, country="USA", occupation="Developer")

# Demonstrating a mix of positional, keyword, and default arguments
print("\nUsing a mix of positional, keyword, and default arguments:")
show_details(name, age, occupation="Engineer")
```

Output:-

```
Enter your name: Shivani
Enter your age: 19\

Using positional arguments:
Name: Shivani
Age: 19\
Country: Unknown
Occupation: Student
```

```
Using keyword arguments:
Name: Shivani
Age: 19\
Country: USA
Occupation: Developer

Using a mix of positional, keyword, and default arguments:
Name: Shivani
Age: 19\
Country: Unknown
Occupation: Engineer
```

Program 10:-Write a program to show variable length argument and its use.

```
def add_numbers(*args):
    total = 0
    for num in args:
        total += num
    return total

def display_info(**kwargs):
    for key, value in kwargs.items():
```

```
    print(f"{key}: {value}")

# Taking input from the user
numbers_input = input("Enter numbers separated by spaces to add: ")
numbers = [int(num) for num in numbers_input.split()]

# Demonstrating variable-length positional arguments (*args)
print("\nUsing variable-length positional arguments (*args):")
result = add_numbers(*numbers)
print(f"Sum of numbers: {result}")

# Taking input from the user for keyword arguments
name = input("Enter your name: ")
age = input("Enter your age: ")
country = input("Enter your country: ")

# Demonstrating variable-length keyword arguments (**kwargs)
print("\nUsing variable-length keyword arguments (**kwargs):")
display_info(name=name, age=age, country=country)
```

Output:-

```
Enter numbers separated by spaces to add: 12 34 56 78 90 48 22 10

Using variable-length positional arguments (*args):
Sum of numbers: 350
Enter your name: Shivani
Enter your age: 19
Enter your country: India

Using variable-length keyword arguments (**kwargs):
name: Shivani
age: 19
country: India
```

Program 11:-Write a lambda/Anonymous function to find bigger number in two given numbers.

```
# Lambda function to find the bigger number
bigger_number = lambda x, y: x if x > y else y

# Taking input from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

# Using the lambda function
result = bigger_number(num1, num2)

print(f"The bigger number is: {result}")
```

Output:-

```
Enter the first number: 10
Enter the second number: 12
The bigger number is: 12.0
```

Program 12:-Create a decorator function to increase the value of a function by 3.

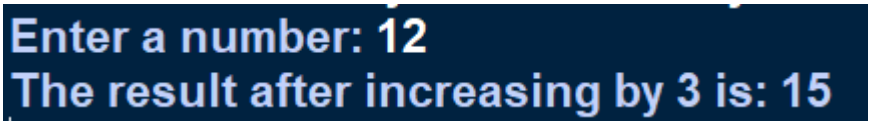
```
# Decorator function to increase the result by 3
def increase_by_three(func):
```

```
    def wrapper(*args, **kwargs):
        # Call the original function and add 3 to its result
        result = func(*args, **kwargs)
        return result + 3
    return wrapper

# Example function to be decorated
@increase_by_three
def get_number():
    return int(input("Enter a number: "))

# Using the decorated function
result = get_number()
print(f"The result after increasing by 3 is: {result}")
```

Output:-



Program 13:-Create a program name “employee.py” and implement the functions DA, HRA, PF, and ITAX. Create another program that uses the function of employee module and calculates gross and net salaries of an employee.

```
import employee
def main():
    # Input basic salary
    basic_salary = float(input("Enter the basic salary: "))

    # Calculate allowances and deductions
    da = employee.calculate_da(basic_salary)
    hra = employee.calculate_hra(basic_salary)
    pf = employee.calculate_pf(basic_salary)

    # Calculate gross salary
    gross_salary = basic_salary + da + hra

    # Calculate income tax
    itax = employee.calculate_itax(gross_salary)

    # Calculate net salary
    net_salary = gross_salary - (pf + itax)

    # Print the results
    print(f"Basic Salary: {basic_salary}")
    print(f"Dearness Allowance (DA): {da}")
    print(f"House Rent Allowance (HRA): {hra}")
    print(f"Provident Fund (PF): {pf}")
    print(f"Income Tax (ITAX): {itax}")
    print(f"Gross Salary: {gross_salary}")
    print(f"Net Salary: {net_salary}")
if __name__ == "__main__":
    main()
```

employee.py:-

```
def calculate_da(basic_salary):
    """Calculate Dearness Allowance (DA)"""
    return basic_salary * 0.10  # Assume DA is 10% of basic salary

def calculate_hra(basic_salary):
    """Calculate House Rent Allowance (HRA)"""
    return basic_salary * 0.20  # Assume HRA is 20% of basic salary

def calculate_pf(basic_salary):
    """Calculate Provident Fund (PF)"""
    return basic_salary * 0.12  # Assume PF is 12% of basic salary

def calculate_itax(gross_salary):
    """Calculate Income Tax (ITAX)"""
    return gross_salary * 0.15  # Assume ITAX is 15% of gross salary
```

Output:-

```
Enter the basic salary: 100000
Basic Salary: 100000.0
Dearness Allowance (DA): 10000.0
House Rent Allowance (HRA): 20000.0
Provident Fund (PF): 12000.0
Income Tax (ITAX): 19500.0
Gross Salary: 130000.0
Net Salary: 98500.0
```

Program 14:-Write a program to create a list using range functions and perform append, update and delete elements operations in it.

```
def main():
    # Create a list using the range function
    start = int(input("Enter the start of the range: "))
    end = int(input("Enter the end of the range: "))
    my_list = list(range(start, end + 1))

    print(f"Initial list: {my_list}")

    # Append an element to the list
    element_to_append = int(input("Enter an element to append: "))
    my_list.append(element_to_append)
    print(f"List after appending {element_to_append}: {my_list}")

    # Update an element in the list
    index_to_update = int(input("Enter the index of the element to update: "))
    new_value = int(input("Enter the new value: "))
    if 0 <= index_to_update < len(my_list):
        my_list[index_to_update] = new_value
        print(f"List after updating index {index_to_update} with {new_value}: {my_list}")
    else:
        print("Invalid index! No update performed.")

    # Delete an element from the list
```

```
    index_to_delete = int(input("Enter the index of the element to delete: "))
    if 0 <= index_to_delete < len(my_list):
        deleted_value = my_list.pop(index_to_delete)
        print(f"List after deleting element at index {index_to_delete} (value {deleted_value}):
{my_list}")
    else:
        print("Invalid index! No deletion performed.")
if __name__ == "__main__":
    main()
```

Output:-

```
Enter the start of the range: 1
Enter the end of the range: 7
Initial list: [1, 2, 3, 4, 5, 6, 7]
Enter an element to append: 9
List after appending 9: [1, 2, 3, 4, 5, 6, 7, 9]
Enter the index of the element to update: 7
Enter the new value: 8
List after updating index 7 with 8: [1, 2, 3, 4, 5, 6, 7, 8]
Enter the index of the element to delete: 7
List after deleting element at index 7 (value 8): [1, 2, 3, 4, 5, 6, 7]
```

Program 15:-Write a program to combine two List, perform repetition of lists and create cloning of lists. Write a program to combine two List, perform repetition of lists and create cloning of lists.

```
    # Input for the first list
list1 = input("Enter the elements of the first list, separated by space: ").split()

    # Input for the second list
list2 = input("Enter the elements of the second list, separated by space: ").split()

    # Combine the two lists
combined_list = list1 + list2
print(f"Combined list: {combined_list}")

    # Perform repetition of the combined list
repetition_count = int(input("Enter the number of times to repeat the combined list: "))
repeated_list = combined_list * repetition_count
print(f"Repeated list: {repeated_list}")

    # Create a clone of the repeated list
cloned_list = repeated_list[:]
print(f"Cloned list: {cloned_list}")

    # Show that the cloned list is indeed a separate copy
cloned_list.append("new_element")
print(f"Original repeated list after cloning: {repeated_list}")
print(f"Cloned list after appending a new element: {cloned_list}")
```

Output:-

```
Enter the elements of the first list, separated by space: 1 2 3 4 5
Enter the elements of the second list, separated by space: 6 7 8 9 10
Combined list: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
Enter the number of times to repeat the combined list: 2
Repeated list: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
Cloned list: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
Original repeated list after cloning: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
Cloned list after appending a new element: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'new_element']
```

Program 16:-Create a sample list of 7 elements and implement the List methods mentioned: append, insert, copy, extend, count, remove, pop, sort, reverse and clear.

```
def main():
    # Create a sample list with 7 elements provided by the user
    my_list = []
    for i in range(7):
        element = input(f"Enter element {i+1}: ")
        my_list.append(element)

    print(f"Original list: {my_list}")

    # Append an element to the list
    element_to_append = input("Enter an element to append: ")
    my_list.append(element_to_append)
    print(f"List after appending '{element_to_append}': {my_list}")

    # Insert an element at a specific index
    index_to_insert = int(input("Enter the index where you want to insert an element: "))
    element_to_insert = input("Enter the element to insert: ")
    my_list.insert(index_to_insert, element_to_insert)
    print(f"List after inserting '{element_to_insert}' at index {index_to_insert}: {my_list}")

    # Copy the list
    copied_list = my_list.copy()
    print(f"Copied list: {copied_list}")

    # Extend the list by another list
    elements_to_extend = input("Enter elements to extend the list with, separated by space: ").split()
    my_list.extend(elements_to_extend)
    print(f"List after extending with {elements_to_extend}: {my_list}")

    # Count occurrences of an element
    element_to_count = input("Enter an element to count its occurrences: ")
    count_of_element = my_list.count(element_to_count)
    print(f"Count of '{element_to_count}' in the list: {count_of_element}")

    # Remove the first occurrence of an element
    element_to_remove = input("Enter the element to remove: ")
    if element_to_remove in my_list:
        my_list.remove(element_to_remove)
        print(f"List after removing the first occurrence of '{element_to_remove}': {my_list}")
    else:
        print(f"Element '{element_to_remove}' not found in the list.")

    # Pop an element from a specific index
    index_to_pop = int(input("Enter the index of the element to pop: "))
    if 0 <= index_to_pop < len(my_list):
        popped_element = my_list.pop(index_to_pop)
        print(f"List after popping element at index {index_to_pop} (popped element:
{popped_element}): {my_list}")
    else:
```

```
        print("Invalid index! No element popped.")

    # Sort the list
    try:
        my_list.sort()
        print(f"List after sorting: {my_list}")
    except TypeError:
        print("Cannot sort the list because it contains elements of different types.")

    # Reverse the list
    my_list.reverse()
    print(f"List after reversing: {my_list}")

    # Clear the list
    my_list.clear()
    print(f"List after clearing all elements: {my_list}")
if __name__ == "__main__":
    main()
```

Output:-

```
Enter element 1: 12
Enter element 2: 24
Enter element 3: 56
Enter element 4: 34
Enter element 5: 79
Enter element 6: 46
Enter element 7: 10
Original list: ['12', '24', '56', '34', '79', '46', '10']
Enter an element to append: 55
List after appending '55': ['12', '24', '56', '34', '79', '46', '10', '55']
Enter the index where you want to insert an element: 5
Enter the element to insert: 10
List after inserting '10' at index 5: ['12', '24', '56', '34', '79', '10', '46', '10', '55']
Copied list: ['12', '24', '56', '34', '79', '10', '46', '10', '55']
Enter elements to extend the list with, separated by space: 10 20 30
List after extending with ['10', '20', '30']: ['12', '24', '56', '34', '79', '10', '46', '10', '55', '10', '20', '30']
Enter an element to count its occurrences: 10
Count of '10' in the list: 3
Enter the element to remove: 30
List after removing the first occurrence of '30': ['12', '24', '56', '34', '79', '10', '46', '10', '55', '10', '20']
Enter the index of the element to pop: 7
List after popping element at index 7 (popped element: 10): ['12', '24', '56', '34', '79', '10', '46', '55', '10', '20']
List after sorting: ['10', '10', '12', '20', '24', '34', '46', '55', '56', '79']
List after reversing: ['79', '56', '55', '46', '34', '24', '20', '12', '10', '10']
List after clearing all elements: []
```

Program 17:-Write a program to create nested list and display its elements.

```
def main():# Create an empty list to store the nested list
    nested_list = []

    # Input the number of sub-lists
    num_of_sublists = int(input("Enter the number of sub-lists: "))

    for i in range(num_of_sublists):
        # Input elements for each sub-list
        sublist = input(f"Enter the elements for sub-list {i+1}, separated by space: ").split()
        nested_list.append(sublist)

    # Display the nested list
    print("\nNested list:")
    for i, sublist in enumerate(nested_list, start=1):
        print(f"Sub-list {i}: {sublist}")
```

```
    # Display elements of the nested list individually
    print("\nIndividual elements of the nested list:")
    for i, sublist in enumerate(nested_list, start=1):
        for j, element in enumerate(sublist, start=1):
            print(f"Element [{i}][{j}]: {element}")
if __name__ == "__main__":
    main()
```

Output:-

```
Enter the number of sub-lists: 4
Enter the elements for sub-list 1, separated by space: 2
Enter the elements for sub-list 2, separated by space: 6
Enter the elements for sub-list 3, separated by space: 4
Enter the elements for sub-list 4, separated by space: 3

Nested list:
Sub-list 1: ['2']
Sub-list 2: ['6']
Sub-list 3: ['4']
Sub-list 4: ['3']

Individual elements of the nested list:
Element [1][1]: 2
Element [2][1]: 6
Element [3][1]: 4
Element [4][1]: 3
```

Program 18:-Write a program to accept elements in the form of a tuple and display its minimum, maximum, sum and average.

```
def main():
    # Accept elements from the user in the form of a tuple
    elements = input("Enter the elements of the tuple, separated by space: ").split()

    # Convert the input elements to a tuple of integers
    elements_tuple = tuple(map(int, elements))

    # Calculate the minimum, maximum, sum, and average
    minimum = min(elements_tuple)
    maximum = max(elements_tuple)
    total_sum = sum(elements_tuple)
    average = total_sum / len(elements_tuple) if elements_tuple else 0

    # Display the results
    print(f"\nTuple: {elements_tuple}")
    print(f"Minimum: {minimum}")
    print(f"Maximum: {maximum}")
    print(f"Sum: {total_sum}")
    print(f"Average: {average:.2f}")
if __name__ == "__main__":
    main()
```

Output:-

```
Enter the elements of the tuple, separated by space: 12 56 34 29 78 10 69

Tuple: (12, 56, 34, 29, 78, 10, 69)
Minimum: 10
Maximum: 78
Sum: 288
Average: 41.14
```

Program 19:-Create a program to sort tuple with nested tuples.

```
def main():
    # Accept nested tuples from the user
```

```
    nested_tuples_input = input("Enter the nested tuples, each separated by a semicolon (e.g., (3, 4); (1, 2); (5, 0)): ")

    # Convert the input into a tuple of tuples
    nested_tuples = tuple(eval(t.strip()) for t in nested_tuples_input.split(';'))

    # Sort the tuple based on the first element of each nested tuple
    sorted_tuples = tuple(sorted(nested_tuples, key=lambda x: x[0]))

    # Display the sorted tuple
    print(f"\nOriginal nested tuple: {nested_tuples}")
    print(f"Sorted nested tuple: {sorted_tuples}")
if __name__ == "__main__":
    main()
```

Output:-

```
Enter the nested tuples, each separated by a semicolon (e.g., (3, 4); (1, 2); (5, 0)): (2,6);(4,9);(4,1)

Original nested tuple: ((2, 6), (4, 9), (4, 1))
Sorted nested tuple: ((2, 6), (4, 9), (4, 1))
```

Program 20:-Write a program to create a dictionary from the user and display the elements.

```
# Initialize an empty dictionary
my_dict = {}

# Number of key-value pairs to be added
n = int(input("Enter the number of key-value pairs: "))

# Loop to get key-value pairs from the user
for _ in range(n):
    key = input("Enter key: ")
    value = input("Enter value: ")
    my_dict[key] = value

# Display the dictionary
print("The dictionary elements are:")
for key, value in my_dict.items():
    print(f"{key}: {value}")
```

Output:-

```
Enter the number of key-value pairs: 3
Enter key: 1
Enter value: abc
Enter key: 2
Enter value: defg
Enter key: 3
Enter value: hijk
The dictionary elements are:
1: abc
2: defg
3: hijk
```

Program 21:-Create a dictionary that will accept cricket players name and scores in a match. Also we are retrieving runs by entering the player's name.

```
# Initialize an empty dictionary
players_scores = {}

# Number of players
n = int(input("Enter the number of players: "))

# Loop to get player names and scores from the user
for _ in range(n):
    name = input("Enter player name: ")
    score = int(input(f"Enter {name}'s score: "))
    players_scores[name] = score

# Display the dictionary
print("\nThe players and their scores are:")
for name, score in players_scores.items():
    print(f"{name}: {score}")

# Retrieve a player's score
player_name = input("\nEnter the player's name to retrieve their score: ")
score = players_scores.get(player_name, "Player not found")
print(f"{player_name}'s score: {score}")
```

Output:-

```
Enter the number of players: 2
Enter player name: abc
Enter abc's score: 56
Enter player name: xyz
Enter xyz's score: 65
```

```
The players and their scores are:
abc: 56
xyz: 65

Enter the player's name to retrieve their score: abc
abc's score: 56
```

Program 22:-Write a program to convert the elements of two lists into key-value pairs of a dictionary.

```
# Function to get a list of elements from the user
def get_list(prompt):
    return input(prompt).split()

# Get the keys and values from the user
print("Enter the keys separated by spaces:")
keys = get_list("Keys: ")

print("Enter the values separated by spaces:")
values = get_list("Values: ")

# Check if the lengths of the lists match
if len(keys) != len(values):
    print("Error: The number of keys must match the number of values.")
else:
    # Create the dictionary from the lists
    dictionary = dict(zip(keys, values))
```

```
    # Display the dictionary
    print("\nThe dictionary created from the lists is:")
    for key, value in dictionary.items():
        print(f"{key}: {value}")
```

Output:-

```
Enter the keys separated by spaces:
Keys: 1 2 3
Enter the values separated by spaces:
Values: a b c

The dictionary created from the lists is:
1: a
2: b
3: c
```

Program 23:-Create a python function to accept python function as a dictionary and display its elements.

```
import ast
def display_dictionary_elements(dictionary):
    """Display elements of the given dictionary."""
    print("\nThe dictionary elements are:")
    for key, value in dictionary.items():
        print(f"{key}: {value}")

def get_dictionary_from_user():
    """Accept a dictionary from the user and return it."""
    # Prompt user to input a dictionary
    user_input = input("Enter a dictionary (e.g., {'key1': 'value1', 'key2': 'value2'}): ")

    # Convert the string input to a dictionary using ast.literal_eval
    try:
        dictionary = ast.literal_eval(user_input)
        if not isinstance(dictionary, dict):
            raise ValueError("The input is not a valid dictionary.")
    except (SyntaxError, ValueError) as e:
        print(f"Invalid input: {e}")
        return None

    return dictionary

# Get dictionary from the user
user_dict = get_dictionary_from_user()

# Display elements if dictionary is valid
if user_dict is not None:
    display_dictionary_elements(user_dict)
```

Output:-

```
Enter a dictionary (e.g., {'key1': 'value1', 'key2': 'value2'}): {1: 'ab',2:'cd',3:'ef'}

The dictionary elements are:
1: ab
2: cd
3: ef
```