RAMAIAH
Institute of Technology

Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of Computer Science & Engineering

Data Visualization with Python Lab(CSL48)

USN:		**Week #: 01**
Semester: 4	**Section : D**	**Date:**

Faculty Coordinator : Jamuna S Murthy

Instructions:

- **Implement the following programs using python language.**

Topic: Course Introduction, Deep dive into lists, sets, dictionaries, and tuples; Time complexity analysis. Control Structures

Programs:

1. a. Write a Python program to check whether a given number is even or odd.
b. Write a Python program to calculate the factorial of a number using recursion.

2. Write a Python program to remove duplicates from a list while maintaining order.

3. Write a Python program to remove all duplicate elements from a given list using sets.

4. Write a Python program to count the occurrences of each character in a given string using a dictionary.

5. Write a Python program to find the index of an element in a tuple

6. Write a Python program to generate the first N Fibonacci numbers using both recursion and iteration. Compare their time complexity.

7. a. Write a Python program to find the largest of three numbers using if-else statements.
b. Write a Python program to print the multiplication table of a given number.

8. Write a Python program that prints numbers from 1 to 50, but skips multiples of 5 using the `continue` statement.

9. Write a Python program that stops execution if it encounters a negative number in a list using the `break` statement.

10. Write a Python program using the `pass` statement inside a loop.

1a. Even or Odd Check

- **Uses modulus operator (`%`) to determine divisibility.**
- **If `num % 2 == 0`, the number is even, otherwise it's odd.**
- **Concepts Covered:**
    - **Conditional statements (`if-else`).**
    - **Modulus operator (`%`).**

```
num = int(input("Enter a number: "))
if num % 2 == 0:
    print("Even")
else:
    print("Odd")
```

1b. Factorial Using Recursion

- **Recursion is a technique where a function calls itself.**
- **Base case: If `n == 0` or `n == 1`, return `1`.**
- **Recursive case: `factorial(n) = n * factorial(n - 1)`.**
- **Concepts Covered:**
    - **Recursion.**
    - **Base and recursive cases.**

```
def factorial(n):
    return 1 if n == 0 else n * factorial(n - 1)

num = int(input("Enter a number: "))
print("Factorial:", factorial(num))
```

---

2. Removing Duplicates While Maintaining Order

- **Uses an empty list and iterates through the original list.**
- **If an element is not already in the new list, it's added.**
- **Concepts Covered:**
    - **Lists.**
    - **Looping through lists.**
    - **Checking for element existence using `in`.**

```
def remove_duplicates(lst):
    seen = set()
    return [x for x in lst if not (x in seen or seen.add(x))]

lst = [1, 2, 2, 3, 4, 4, 5]
print(remove_duplicates(lst))
```

---

3. Removing Duplicates Using Sets

- **A set is an unordered collection with unique elements.**
- **Convert the list to a set and back to a list.**
- **Concepts Covered:**
    - **Sets (`set()`).**
    - **Type conversion (`list(set())`).**

```
lst = [1, 2, 2, 3, 4, 4, 5]
unique_lst = list(set(lst))
print(unique_lst)
```

---

4. Counting Character Occurrences in a String

- **Uses a dictionary to store characters as keys and their occurrences as values.**
- **Iterates through the string, updating counts.**
- **Concepts Covered:**
    - **Dictionaries.**
    - **Looping through strings.**
    - **Dictionary key-value updates.**

```
def char_count(s):
    counts = {}
    for char in s:
        counts[char] = counts.get(char, 0) + 1
    return counts

s = input("Enter a string: ")
print(char_count(s))
```

---

5. Finding Index of an Element in a Tuple

- **Uses the `.index()` method of tuples.**
- **Concepts Covered:**
  - **Tuples (immutable sequences).**
  - **The `.index()` method.**

```
tup = (10, 20, 30, 40, 50)
element = 30
print("Index of", element, ":", tup.index(element))
```

---

6. Fibonacci Numbers (Recursion vs. Iteration)

Recursive Approach:

- **Base case: `fib(0) = 0`, `fib(1) = 1`.**
- **Recursive formula: `fib(n) = fib(n-1) + fib(n-2)`.**
- **Time Complexity: Exponential $O(2^n)$ (inefficient).**

Iterative Approach:

- **Uses a loop to generate Fibonacci numbers.**
- **More efficient, runs in O(n) time.**

Concepts Covered:

- **Recursion vs. Iteration.**
- **Time Complexity Analysis.**

```
import time

def fibonacci_recursive(n):
    if n <= 1:
        return n
    return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2)

def fibonacci_iterative(n):
    fib_series = [0, 1]
    for _ in range(n - 2):
        fib_series.append(fib_series[-1] + fib_series[-2])
    return fib_series[:n]
```

```
n = int(input("Enter N: "))

start = time.time()
print("Recursive Fibonacci:", [fibonacci_recursive(i) for i in range(n)])
print("Recursive time:", time.time() - start)

start = time.time()
print("Iterative Fibonacci:", fibonacci_iterative(n))
print("Iterative time:", time.time() - start)
```

7a. Finding the Largest of Three Numbers

- **Uses `if-elif-else` to compare three numbers.**
- **Concepts Covered:**
  - **Conditional statements.**
  - **Comparisons (`>`, `<`).**

```
a, b, c = map(int, input("Enter three numbers: ").split())
largest = a if a > b and a > c else b if b > c else c
print("Largest:", largest)
```

7b. Printing Multiplication Table

- **Uses a loop to multiply a number from `1` to `10`.**
- **Concepts Covered:**
  - **Loops (`for` loop).**
  - **String formatting.**

```
num = int(input("Enter a number: "))
for i in range(1, 11):
    print(f"{num} x {i} = {num * i}")
```

---

8. Skipping Multiples of 5 Using `continue`

- **Loops through numbers from 1 to 50.**
- **Uses `if num % 5 == 0: continue` to skip multiples of 5.**
- **Concepts Covered:**
  - **`continue` statement (skips iteration).**
  - **Modulus operator.**

```
for i in range(1, 51):
    if i % 5 == 0:
        continue
    print(i, end=" ")
```

---

9. Stopping Execution with `break` on a Negative Number

- **Iterates through a list and checks for negative numbers.**
- **If a negative number is found, `break` stops execution.**
- **Concepts Covered:**
    - `break` **statement (exits loop).**
    - **Conditional checks.**

```
lst = [10, 20, -5, 30, 40]
for num in lst:
    if num < 0:
        print("Negative number encountered! Stopping execution.")
        break
    print(num)
```

---

10. Using `pass` Inside a Loop

- `pass` **is a placeholder that does nothing.**
- **Used when a loop or function needs to exist syntactically but does not perform any operation.**
- **Concepts Covered:**
    - `pass` **statement (used for placeholders).**

```
for i in range(5):
    if i == 2:
        pass  # Placeholder for future code
    print(i)
```