

Python Programming Lab Manual - 2023 Syllabus - BCA

1. Write a Python program to demonstrate the use of various operators (arithmetic, relational, logical, bitwise) on variables.
2. Write a Python program to perform the string operations.
3. Write a Python program to input a number and check whether it is odd or even using an if-else conditional statement.
4. Define a user-defined module containing a function to calculate the factorial of a number. Import and call the function from another script.
5. Create a Python script that takes a sentence as input and counts the number of vowels and consonants.
6. Write a Python program to demonstrate list operations like insertion, deletion, slicing, and concatenation.
7. Implement a Python program to demonstrate set operations like union, intersection, and difference.
8. Write a Python program to create a tuple, access its elements, and perform tuple slicing. Demonstrate immutability of tuples.
9. Write a recursive Python function to find the greatest common divisor (GCD) of two numbers.
10. Write a Python program to read and write data to a text file. The program should take user input and store it in the file, then read it back and display it.
11. Write a program to implement a Python script that demonstrates the use of try-except to handle division by zero errors and other common exceptions.
12. Write a Python program to define a Student class with attributes name, age, and grade. Create objects of this class and display their details.
13. Tkinter Label and Button: Create a Python GUI using tkinter that displays a label and a button. When the button is clicked, change the label's text.
14. Develop a GUI application in Python using tkinter that takes a user's name as input in an entry widget and displays it using a label when a button is clicked.
15. Radio Buttons and Check Buttons: Create a Python GUI application using tkinter with a set of radio buttons to select a gender and check buttons to select hobbies. Display the selected options when a button is pressed.
16. Write a Python program that demonstrates client-server communication using sockets on the same machine. The client sends a message to the server, and the server responds with a confirmation message.
17. Database Operations - Insertion and Retrieval: Develop a Python script that creates a database and table using SQL. Insert data into the table and retrieve it using a SELECT statement.

1. Write a Python program to demonstrate the use of various operators (arithmetic, relational, logical, bitwise) on variables.

```
# Declare two integer variables
```

```
a = 10
```

```
b = 5
```

```
print("=== Arithmetic Operators ===")
```

```
print(f"{a} + {b} = {a + b}")    # Addition
```

```
print(f"{a} - {b} = {a - b}")    # Subtraction
```

```
print(f"{a} * {b} = {a * b}")    # Multiplication
```

```
print(f"{a} / {b} = {a / b}")    # Division (float)
```

```
print(f"{a} // {b} = {a // b}")  # Floor Division
```

```
print(f"{a} % {b} = {a % b}")    # Modulus
```

```
print(f"{a} ** {b} = {a ** b}")  # Exponentiation
```

```
print("\n=== Relational (Comparison) Operators ===")
```

```
print(f"{a} == {b} -> {a == b}") # Equal to
```

```
print(f"{a} != {b} -> {a != b}") # Not equal to
```

```
print(f"{a} > {b} -> {a > b}")   # Greater than
```

```
print(f"{a} < {b} -> {a < b}")   # Less than
```

```
print(f"{a} >= {b} -> {a >= b}") # Greater than or equal to
```

```
print(f"{a} <= {b} -> {a <= b}") # Less than or equal to
```

```
print("\n=== Logical Operators ===")
```

```
x = True
```

```
y = False
```

```
print(f"{x} and {y} -> {x and y}") # Logical AND
```

```
print(f"{x} or {y} -> {x or y}")   # Logical OR
```

```
print(f"not {x} -> {not x}")        # Logical NOT
```

```
print("\n=== Bitwise Operators ===")
```

```
print(f"{a} & {b} = {a & b}")       # Bitwise AND
```

```
print(f"{a} | {b} = {a | b}")       # Bitwise OR
```

```
print(f"{a} ^ {b} = {a ^ b}")       # Bitwise XOR
```

```
print(f"~{a} = {~a}")               # Bitwise NOT
```

```
print(f"{a} << 1 = {a << 1}")       # Left Shift
```

```
print(f"{a} >> 1 = {a >> 1}")       # Right Shift
```

Output:

```
=== Arithmetic Operators ===
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2.0
10 // 5 = 2
10 % 5 = 0
10 ** 5 = 100000

=== Relational (Comparison) Operators ===
10 == 5 -> False
10 != 5 -> True
10 > 5 -> True
10 < 5 -> False
10 >= 5 -> True
10 <= 5 -> False

=== Logical Operators ===
True and False -> False
True or False -> True
not True -> False

=== Bitwise Operators ===
10 & 5 = 0
10 | 5 = 15
10 ^ 5 = 15
~10 = -11
10 << 1 = 20
10 >> 1 = 5
```

2. Write a Python program to perform the string operations.

```
# Define two strings
str1 = "Hello"
str2 = "World"
print("=== String Operations ===")
# Concatenation
print(f"Concatenation: {str1} + {str2} = {str1 + str2}")
# Repetition
print(f"Repetition: {str1} * 3 = {str1 * 3}")
# Length of the string
print(f"Length of str1: len('{str1}') = {len(str1)}")
# Indexing
print(f"First character of str1: {str1[0]}")
print(f"Last character of str2: {str2[-1]}")
```

```
# Slicing
print(f"Slicing str1[1:4]: {str1[1:4]}")
# Membership
print(f"'lo' in str1 -> {'lo' in str1}")
print(f"'z' not in str2 -> {'z' not in str2}")
# Case Conversion
print(f"Uppercase: {str1.upper()}")
print(f"Lowercase: {str2.lower()}")
# Replace
print(f"Replace 'l' with 'x' in str1: {str1.replace('l', 'x')}")
# Strip
str3 = "  spaced string  "
print(f"Original: '{str3}'")
print(f"After strip(): '{str3.strip()}'")
# Split and Join
csv = "apple,banana,cherry"
fruits = csv.split(",")
print(f"Split: {csv} -> {fruits}")
joined = "-".join(fruits)
print(f"Join with '-': {joined}")
# String formatting
name = "Alice"
age = 25
print(f"Formatted string: My name is {name} and I am {age} years old.")
# Find and Count
sentence = "banana is a banana fruit"
print(f"Find 'banana': {sentence.find('banana')}")
print(f"Count 'banana': {sentence.count('banana')}")
```

Output:

```
=== String Operations ===
Concatenation: Hello + World = HelloWorld
Repetition: Hello * 3 = HelloHelloHello
Length of str1: len('Hello') = 5
First character of str1: H
Last character of str2: d
Slicing str1[1:4]: ell
'lo' in str1 -> True
'z' not in str2 -> True
Uppercase: HELLO
Lowercase: world
Replace 'l' with 'x' in str1: Hexxo
Original: '  spaced string '
After strip(): 'spaced string'
Split: apple,banana,cherry -> ['apple', 'banana', 'cherry']
Join with '-': apple-banana-cherry
Formatted string: My name is Alice and I am 25 years old.
Find 'banana': 0
Count 'banana': 2
```

3. Write a Python program to input a number and check whether it is odd or even using an if-else conditional statement.

```
# Input a number from the user
num = int(input("Enter a number: "))
# Check whether the number is even or odd
if num % 2 == 0:
    print(f"{num} is an Even number.")
else:
    print(f"{num} is an Odd number.")
```

Output:

```
Enter a number: 45
45 is an Odd number.
```

4. Define a user-defined module containing a function to calculate the factorial of a number. Import and call the function from another script.

[mymath.py](#)

```
# mymath.py
def factorial(n):
    """Recursive function to calculate factorial of n"""
    if n == 0 or n == 1:
```

```
        return 1
    else:
        return n * factorial(n - 1)
```

[main.py](#)

```
# main.py
import mymath # Importing the user-defined module
# Input from user
num = int(input("Enter a number to find its factorial: "))
# Calling the factorial function from mymath module
result = mymath.factorial(num)
# Display result
print(f"The factorial of {num} is {result}")
```

Output:

```
Enter a number to find its factorial: 6
The factorial of 6 is 720
```

5. Create a Python script that takes a sentence as input and counts the number of vowels and Consonants.

```
# Input a sentence from the user
sentence = input("Enter a sentence: ")
# Initialize counters
vowel_count = 0
consonant_count = 0
# Define vowels
vowels = "aeiouAEIOU"
# Iterate through each character in the sentence
for char in sentence:
    if char.isalpha(): # Check if the character is a letter
        if char in vowels:
            vowel_count += 1
        else:
            consonant_count += 1
# Output the results
print(f"Number of vowels: {vowel_count}")
print(f"Number of consonants: {consonant_count}")
```

Output:

```
Enter a sentence: Hajiram Beevi
Number of vowels: 6
Number of consonants: 6
```

6. Write a Python program to demonstrate list operations like insertion, deletion, slicing, and Concatenation.

```
# Initial list
fruits = ['apple', 'banana', 'cherry']
print("Original list:", fruits)
# Insertion
fruits.append('orange') # Add at the end
print("After appending 'orange':", fruits)
fruits.insert(1, 'mango') # Insert at index 1
print("After inserting 'mango' at index 1:", fruits)
# Deletion
fruits.remove('banana') # Remove by value
print("After removing 'banana':", fruits)
removed_item = fruits.pop(2) # Remove by index
print(f"After popping item at index 2 ({removed_item}):", fruits)
# Slicing
print("Slicing fruits[1:3]:", fruits[1:3]) # Slice from index 1 to 2
# Concatenation
vegetables = ['carrot', 'potato']
combined = fruits + vegetables
print("Concatenated list (fruits + vegetables):", combined)
```

Output:

```
Original list: ['apple', 'banana', 'cherry']
After appending 'orange': ['apple', 'banana', 'cherry', 'orange']
After inserting 'mango' at index 1: ['apple', 'mango', 'banana', 'cherry', 'orange']
After removing 'banana': ['apple', 'mango', 'cherry', 'orange']
After popping item at index 2 ('cherry'): ['apple', 'mango', 'orange']
Slicing fruits[1:3]: ['mango', 'orange']
Concatenated list (fruits + vegetables): ['apple', 'mango', 'orange', 'carrot', 'potato']
```

7. Implement a Python program to demonstrate set operations like union, intersection, and Difference.

```
# Define two sets
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
print("Set 1:", set1)
print("Set 2:", set2)
# Union of sets
union_set = set1.union(set2)
print("\nUnion of Set1 and Set2:", union_set)
# Intersection of sets
intersection_set = set1.intersection(set2)
print("Intersection of Set1 and Set2:", intersection_set)
# Difference of sets
difference_set1 = set1.difference(set2)
difference_set2 = set2.difference(set1)
print("Difference of Set1 - Set2:", difference_set1)
print("Difference of Set2 - Set1:", difference_set2)
```

Output:

```
Set 1: {1, 2, 3, 4, 5}
Set 2: {4, 5, 6, 7, 8}

Union of Set1 and Set2: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection of Set1 and Set2: {4, 5}
Difference of Set1 - Set2: {1, 2, 3}
Difference of Set2 - Set1: {8, 6, 7}
```

8. Write a Python program to create a tuple, access its elements, and perform tuple slicing. Demonstrate immutability of tuples.

```
# Creating a tuple
my_tuple = ('apple', 'banana', 'cherry', 'date', 'elderberry')
print("Original tuple:", my_tuple)
# Accessing elements
print("\nAccessing elements:")
print("First element:", my_tuple[0])
print("Last element:", my_tuple[-1])
# Tuple slicing
print("\nTuple slicing:")
print("Elements from index 1 to 3:", my_tuple[1:4])
print("First three elements:", my_tuple[:3])
```



```
print("Last two elements:", my_tuple[-2:])
# Demonstrate immutability
print("\nDemonstrating immutability:")
try:
    my_tuple[1] = 'blueberry' # Attempt to modify an element
except TypeError as e:
    print("Error:", e)
```

Output:

```
Original tuple: ('apple', 'banana', 'cherry', 'date', 'elderberry')

Accessing elements:
First element: apple
Last element: elderberry

Tuple slicing:
Elements from index 1 to 3: ('banana', 'cherry', 'date')
First three elements: ('apple', 'banana', 'cherry')
Last two elements: ('date', 'elderberry')

Demonstrating immutability:
Error: 'tuple' object does not support item assignment
```

9. Write a recursive Python function to find the greatest common divisor (GCD) of two numbers.

```
def gcd(a, b):
    # Base case
    if b == 0:
        return a
    else:
        # Recursive call
        return gcd(b, a % b)

# Example usage
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
result = gcd(num1, num2)
print(f"The GCD of {num1} and {num2} is {result}")
```

Output:

```
Enter first number: 4
Enter second number: 78
The GCD of 4 and 78 is 2
```

10. Write a Python program to read and write data to a text file. The program should take user input and store it in the file, then read it back and display it.

```
# File name to work with
filename = "/content/user_data.txt"
# Take input from the user
user_input = input("Enter some text to store in the file: ")
# Write the input to a text file
with open(filename, "w") as file:
    file.write(user_input)
print(f"\nData written to {filename} successfully.")
# Read the content back from the file
with open(filename, "r") as file:
    content = file.read()
# Display the content
print("\nContent read from the file:")
print(content)
```

Output:

```
Enter some text to store in the file: Hello, Mritikka how are you?

Data written to /content/user_data.txt successfully.

Content read from the file:
Hello, Mritikka how are you?
```

11. Write a program to implement a Python script that demonstrates the use of try-except to handle division by zero errors and other common exceptions.

```
try:
    # Take two numbers from the user
    num1 = float(input("Enter the numerator: "))
    num2 = float(input("Enter the denominator: "))
    # Perform division
    result = num1 / num2
except ZeroDivisionError:
```

```
print("Error: Division by zero is not allowed.")
except ValueError:
    print("Error: Please enter valid numbers.")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
else:
    print(f"The result of division is: {result}")
finally:
    print("Program execution completed.")
```

Output:

Case 1:

```
Enter the numerator: 5
Enter the denominator: 0
Error: Division by zero is not allowed.
Program execution completed.
```

Case 2:

```
Enter the numerator: ten
Error: Please enter valid numbers.
Program execution completed.
```

Case 3:

```
Enter the numerator: 5
Enter the denominator: 2
The result of division is: 2.5
Program execution completed.
```

12. Write a Python program to define a Student class with attributes name, age, and grade. Create objects of this class and display their details.

```
# Define the Student class
class Student:
    def __init__(self, name, age, grade):
        self.name = name    # Student's name
        self.age = age      # Student's age
        self.grade = grade  # Student's grade
    # Method to display student details
    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")
        print(f"Grade: {self.grade}")
```

```
print("-" * 20)
# Create Student objects
student1 = Student("Alice", 15, "10th")
student2 = Student("Bob", 16, "11th")
student3 = Student("Charlie", 14, "9th")
# Display their details
print("Student Details:\n")
student1.display_details()
student2.display_details()
student3.display_details()
```

Output:

```
Student Details:

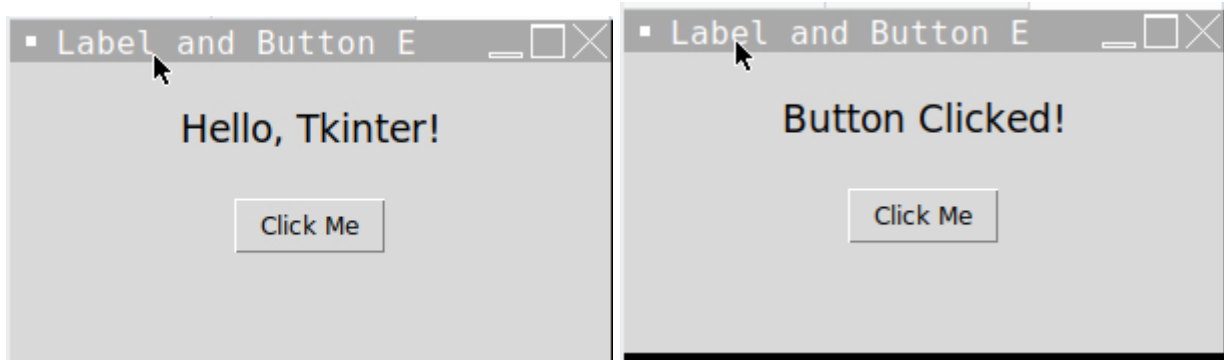
Name: Alice
Age: 15
Grade: 10th
-----
Name: Bob
Age: 16
Grade: 11th
-----
Name: Charlie
Age: 14
Grade: 9th
-----
```

13. Tkinter Label and Button: Create a Python GUI using tkinter that displays a label and a button. When the button is clicked, change the label's text.

```
import tkinter as tk
# Function to change the label text
def change_text():
    label.config(text="Button Clicked!")
# Create the main window
window = tk.Tk()
window.title("Label and Button Example")
window.geometry("300x150")
# Create a label
label = tk.Label(window, text="Hello, Tkinter!", font=("Arial", 14))
label.pack(pady=20)
# Create a button
```

```
button = tk.Button(window, text="Click Me", command=change_text)
button.pack()
# Run the application
window.mainloop()
```

Output:

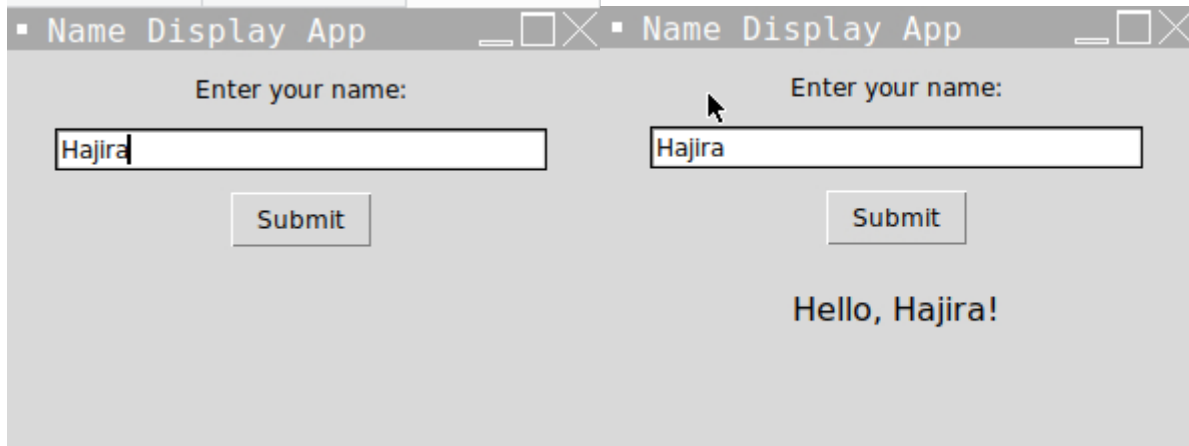


14. Develop a GUI application in Python using tkinter that takes a user's name as input in an entry widget and displays it using a label when a button is clicked.

```
import tkinter as tk
# Function to display the name
def display_name():
    name = entry.get() # Get text from entry widget
    label_output.config(text=f"Hello, {name}!") # Update label
# Create the main application window
window = tk.Tk()
window.title("Name Display App")
window.geometry("300x200")
# Create a label
label_prompt = tk.Label(window, text="Enter your name:")
label_prompt.pack(pady=10)
# Create an entry widget for name input
entry = tk.Entry(window, width=30)
entry.pack()
# Create a button to display the name
button = tk.Button(window, text="Submit", command=display_name)
button.pack(pady=10)
# Label to display the output
label_output = tk.Label(window, text="", font=("Arial", 12))
label_output.pack(pady=10)
```

```
# Run the application
window.mainloop()
```

Output:

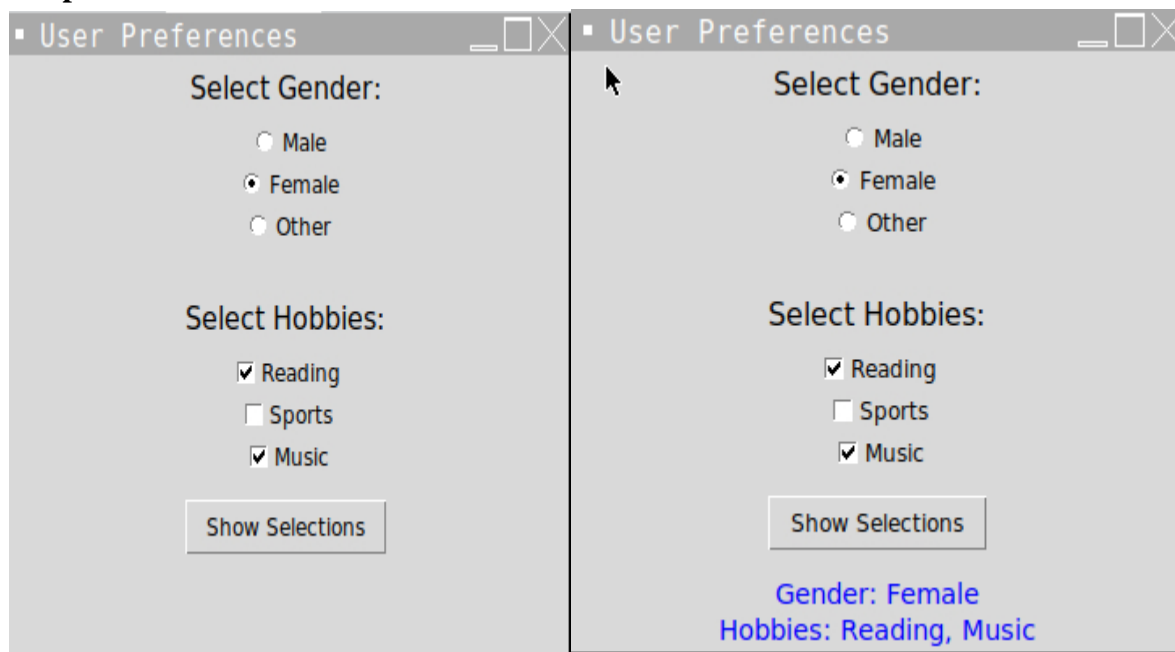


15. Radio Buttons and Check Buttons: Create a Python GUI application using tkinter with a set of radio buttons to select a gender and check buttons to select hobbies. Display the selected options when a button is pressed.

```
import tkinter as tk
def show_selection():
    selected_gender = gender.get()
    selected_hobbies = []
    if hobby_reading.get():
        selected_hobbies.append("Reading")
    if hobby_sports.get():
        selected_hobbies.append("Sports")
    if hobby_music.get():
        selected_hobbies.append("Music")
    result = f"Gender: {selected_gender}\nHobbies: {' '.join(selected_hobbies) if
selected_hobbies else 'None'}"
    output_label.config(text=result)
# Create main window
window = tk.Tk()
window.title("User Preferences")
window.geometry("350x300")
# Gender selection (Radio Buttons)
tk.Label(window, text="Select Gender:", font=('Arial', 12)).pack(pady=5)
gender = tk.StringVar(value="Male") # default
tk.Radiobutton(window, text="Male", variable=gender, value="Male").pack()
```

```
tk.Radiobutton(window, text="Female", variable=gender, value="Female").pack()
tk.Radiobutton(window, text="Other", variable=gender, value="Other").pack()
# Hobby selection (Check Buttons)
tk.Label(window, text="\nSelect Hobbies:", font=('Arial', 12)).pack(pady=5)
hobby_reading = tk.BooleanVar()
hobby_sports = tk.BooleanVar()
hobby_music = tk.BooleanVar()
tk.Checkbutton(window, text="Reading", variable=hobby_reading).pack()
tk.Checkbutton(window, text="Sports", variable=hobby_sports).pack()
tk.Checkbutton(window, text="Music", variable=hobby_music).pack()
# Submit button
tk.Button(window, text="Show Selections", command=show_selection).pack(pady=10)
# Output label
output_label = tk.Label(window, text="", font=("Arial", 11), fg="blue")
output_label.pack()
# Start GUI loop
window.mainloop()
```

Output:



16. Write a Python program that demonstrates client-server communication using sockets on the same machine. The client sends a message to the server, and the server responds with a confirmation message.

```
import socket
import threading
import time
def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('localhost', 12345))
    server_socket.listen(1)
    print("Server is listening...")
    conn, addr = server_socket.accept()
    print("Server: Connected to", addr)
    data = conn.recv(1024).decode()
    print("Server received:", data)
    conn.send("Server received your message!".encode())
    conn.close()
    server_socket.close()
def start_client():
    time.sleep(1) # Wait a moment for the server to be ready
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('localhost', 12345))
    client_socket.send("Hello Server!".encode())
    response = client_socket.recv(1024).decode()
    print("Client received:", response)
    client_socket.close()
# Run server and client in separate threads
threading.Thread(target=start_server).start()
threading.Thread(target=start_client).start()
```

Output:

```
Server is listening...
Server: Connected to ('127.0.0.1', 55908)
Server received: Hello Server!
Client received: Server received your message!
```

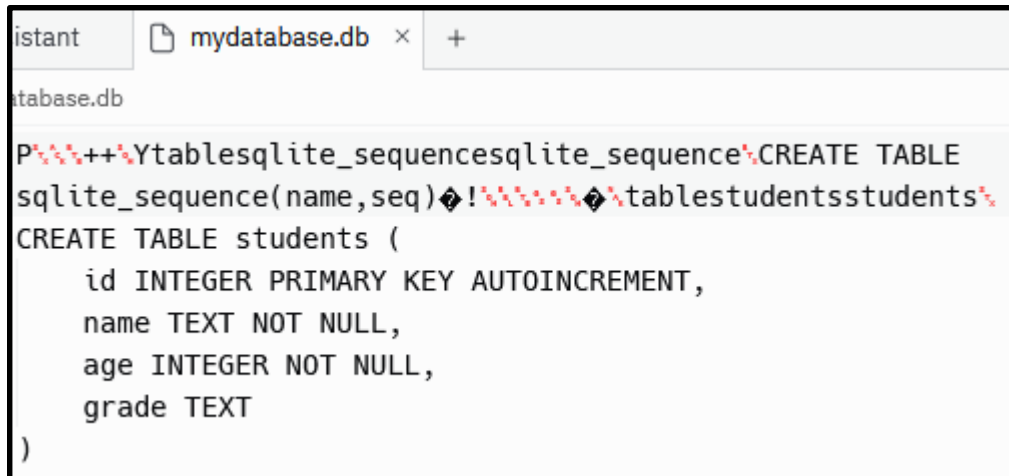
17. Database Operations - Insertion and Retrieval: Develop a Python script that creates a database and table using SQL. Insert data into the table and retrieve it using a SELECT statement.

```
import sqlite3
# Connect to (or create) a SQLite database file
```



```
conn = sqlite3.connect('mydatabase.db') # creates a file named 'mydatabase.db'
cursor = conn.cursor()
# Create a table
cursor.execute("""
CREATE TABLE IF NOT EXISTS students (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    age INTEGER NOT NULL,
    grade TEXT
)
""")
# Insert data into the table
cursor.execute("INSERT INTO students (name, age, grade) VALUES (?, ?, ?)", ("Alice", 20, "A"))
cursor.execute("INSERT INTO students (name, age, grade) VALUES (?, ?, ?)", ("Bob", 22, "B"))
#Commit the transaction
conn.commit()
# Retrieve and display the data
cursor.execute("SELECT * FROM students")
rows = cursor.fetchall()
print("ID | Name | Age | Grade")
for row in rows:
    print(row[0], "|", row[1], "|", row[2], "|", row[3])
# Close the connection
conn.close()
```

Output:



The screenshot shows a database management interface with a tab labeled 'mydatabase.db'. The SQL editor contains the following code:

```
CREATE TABLE
sqlite_sequence(name,seq)
CREATE TABLE students (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    age INTEGER NOT NULL,
    grade TEXT
)
```

ID	Name	Age	Grade
1	Alice	20	A
2	Bob	22	B