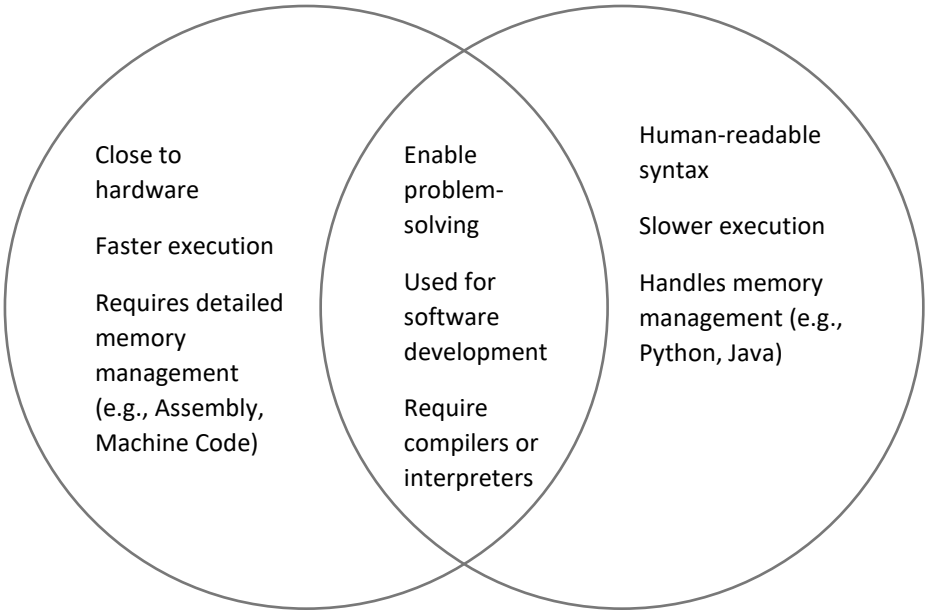

2. What is the primary difference between low-level and high-level programming languages in terms of abstraction from hardware?

    - Low-level programming languages have minimal abstraction from hardware and operate closely with the system's architecture, while high-level programming languages provide a higher level of abstraction, focusing on ease of use and human readability.

3. What languages would typically be found in the overlapping region of a Venn diagram comparing low-level and high-level languages?
    - C and C++ are examples of languages in the overlapping region as they allow for both hardware-level control (low-level features) and abstract, user-friendly coding (high-level features).

4. What is a key feature of high-level programming languages?
    - High-level programming languages prioritize human-readable syntax and automated memory management, simplifying complex coding tasks.
5. Examples of Programming Languages:
    - Low-Level Languages:
        i. Assembly
        ii. Machine Code
        iii. VHDL
        iv. Verilog
        v. Forth
    - High-Level Languages:
        i. Python
        ii. Java
        iii. C#
        iv. Ruby
        v. JavaScript

Assignment 1: “Web Developer”

1. **The roles of back-end, front-end, and full-stack developers differ primarily in the areas of web development they focus on:**
   - **Front-End Developers:**
     Focus on the client-side, creating the visual and interactive elements of a website or app. They work with technologies like HTML, CSS, and JavaScript to ensure a user-friendly and responsive interface.
   - **Back-End Developers:**
     Handle the server-side, focusing on databases, servers, and application logic. They use languages like Python, Java, PHP, or frameworks like Node.js to ensure the website or app functions properly behind the scenes.
   - **Full-Stack Developers:**
     Skilled in both front-end and back-end development, capable of handling the entire web development process, from designing the user interface to managing databases and server-side logic.
2. To build a web application, plan features, choose the tech stack, set up the environment, develop the front-end and back-end, integrate and test, deploy, and maintain the app.
3. Communication between the front-end and back-end is done using asynchronous methods like AJAX, Fetch, or Axios, with RESTful APIs or GraphQL.
4. Common challenges include integration (solved with clear API contracts), CORS (enabled on the back-end), and authentication (using JWT or session cookies).
5. To improve performance, reduce HTTP requests, optimize images, use lazy loading, minify files, implement caching, use a CDN, and load non-essential scripts asynchronously.

Assignment 2: “JavaScript Function”

1.

```
// Function definition for greetUser with default value
function greetUser(name = "Guest") {
  console.log("Hello, " + name + "!");
}

// Example of calling the function with and without a name
greetUser("Alice"); // Output: Hello, Alice!
greetUser(); // Output: Hello, Guest!
```

2. **What happens if the name argument is not provided when calling the function?**
   If the name argument is not provided when calling the greetUser function, it will result in undefined being used as the value for name. This would make the greeting message display as "Hello, undefined!".
3. **Can you modify the function to display a default message if no name is given?**
   Yes, we can modify the function to set a default value for name if no argument is provided. Here's an updated version of the function:

Assignment 3: "Application Programming Interface”

1. What is API

- **Definition**:
  An API (Application Programming Interface) is a set of rules and protocols that allow different software applications to communicate with each other.
- **Functionality**:
  APIs enable the integration of external systems and services, allowing data exchange and enabling features like payment processing, social media sharing, and accessing databases.
- **Using of API**:
  APIs are used to request data, send information, or interact with external services. For example, a weather app using a weather API to fetch real-time data.
- **Process**:
  The process involves making an API request (usually an HTTP request) to a server with parameters, and the server processes it and returns the relevant data or performs an action.
- **Response**:
  The response is typically in formats like JSON or XML. It contains the data requested or an error message if the request failed.
- **Integration**:
  API integration involves embedding external services or data into an application by calling the API and handling the response within the app, allowing seamless functionality.

2. Write a Simple Example of Making an API Request
Here:

```
const apiUrl = 'https://jsonplaceholder.typicode.com/posts';

function fetchData() {
  fetch(apiUrl)
    .then(response => response.json())  // Parse the JSON response
    .then(data => {
      console.log(data);  // Output the data to the console
    })
    .catch(error => {
      console.error('Error fetching data:', error);  // Handle any errors
    });
}

fetchData();
```

Assignment 4: “JavaScript Scope”

1. Global Scope
Variables declared outside any function or block are in the global scope, meaning they can be accessed from anywhere in the code.

2. Function Scope
Variables declared inside a function are scoped to that function and are not accessible outside of it.

3. Block Scope
Variables declared with let or const inside a block (e.g., inside a loop or if statement) are scoped to that block and are not accessible outside of it.
4. Lexical Scope
Lexical scope refers to the fact that a function’s scope is determined by its location in the code, meaning inner functions have access to variables in their outer functions.

5. **What is the difference between Global Scope and Local Scope?**
Global scope refers to variables that are accessible from anywhere in the code, whereas local scope (either function or block) limits variable access to within specific functions or blocks.

6. **What is the scope of a variable declared with let or const inside a loop or block?**
Variables declared with let or const inside a loop or block are block-scoped, meaning they can only be accessed within that block.

7. **Can a variable declared with var inside a function be accessed outside the function?**
No, a variable declared with var inside a function is function-scoped, so it cannot be accessed outside the function.

8. **What is lexical scoping, and how does it affect inner functions?**
Lexical scoping means that an inner function has access to variables of its outer (enclosing) functions based on the code structure, not where the function is called.

9. **What happens if you try to access a variable declared with let or const outside its block?**
If you try to access a variable declared with let or const outside its block, you will get a **ReferenceError** because these variables are block-scoped.

10. **How does JavaScript handle variable hoisting with var, let, and const?**

- **var**: Variables declared with var are hoisted to the top of their scope and initialized with undefined.
- **let and const**: Variables declared with let and const are hoisted, but they are not initialized. Accessing them before their declaration results in a **ReferenceError** due to the **temporal dead zone**.