Institute of Computer Technology
B. Tech Computer Science and Engineering
Sub: (2CSE410) FRONT END TECHNOLOGIES

NAME: SUJAL SUTHAR
SEM: CSE 3-B (BATCH 44)
ER NO. : 23162581026

Practical - 3

AIM: Understanding primitive and non-primitive data types, the concept of arrays and objects in JavaScript, and some of their pre-defined methods. Learning about anonymous and arrow functions in javascript.

Tools Used: Code editor (VS code) and web browser (Google Chrome).

Theory:

❖ **Theory of Arrays:**

In JavaScript, an array is a data structure that allows you to store and manipulate a collection of elements. Arrays can hold elements of any data type, including numbers, strings, objects, and other arrays. JavaScript provides a variety of methods for working with arrays, allowing you to perform operations such as adding or removing elements, iterating over the array, and transforming the array in various ways:

- push(): Adds one or more elements to the end of an array.
- pop(): Removes the last element from the end of an array.
- unshift(): Adds one or more elements to the beginning of an array.
- shift(): Removes the first element from the beginning of an array.

- concat() : Concatenates two or more arrays, creating a new array.
- indexOf(): Returns the index of the first occurrence of a specified element in an array.
- lastIndexOf(): Returns the index of the last occurrence of a specified element in an array.
- includes(): Determines whether an array includes a certain element, returning a boolean value
- reverse(): Reverses the order of the elements in an array.
- splice(): Changes the contents of an array by removing or replacing existing elements and/or adding new elements in place.● slice(): Returns a portion of an array, creating a new array without modifying the original ● filter(): Creates a new array with elements that pass a test implemented by a provided function
- reduce(): Applies a function against an accumulator and each element in the array (from left to right) to reduce it to a single value
- map() : Creates a new array by applying a function to each element of an existing array
- forEach() :Executes a provided function once for each array element.

❖ Theory of Objects :

In JavaScript, an object is a fundamental data type that allows you to store and organize data in key-value pairs. Objects can represent real-world entities and are used to model more complex data structures than arrays. Here are some key concepts related to objects in JavaScript:

❖ Object Declaration:

➢ Objects are created using curly braces {}. Properties and their values are defined inside the braces as key-value pairs.

❖ Accessing Object Properties:

➢ Object properties can be accessed using dot notation (object.property) or square bracket notation (object['property']).

- ❖ Adding and Modifying Properties:
  - ➢ Properties can be added or modified after an object is created.
- ❖ Object Methods:
  - ➢ Functions can be assigned as values to object properties, creating object methods.
- ❖ Object Iteration:
  - ➢ The for...in loop is used to iterate over the properties of an object.
- ❖ Object Methods:
  - ➢ The Object global object provides various methods for working with objects, such as Object.keys(), Object.values(), and Object.entries().

- **Theory of functions:**

  In JavaScript, a function is a block of reusable code that performs a specific task or set of tasks. Functions allow you to organize your code, make it more modular, and avoid repetition. Here's a basic overview of defining and using functions in JavaScript

- Function Declaration: You can declare a function using the function keyword, followed by the function name, a list of parameters enclosed in parentheses, and a block of code enclosed in curly braces.
- Function Invocation: To execute or call a function, you simply use its name followed by parentheses. You can pass arguments (values) to the function if it expects parameters.
- Return Statement: Functions can also return a value using the return statement. The function stops execution when it encounters return, and the specified value is passed back to the calling code.
- Function Expression: Functions can also be assigned to variables, creating what's known as a function expression.
- Arrow Functions (ES6+): Arrow functions provide a concise syntax for writing functions, especially useful for short, one-line functions.

Anonymous Functions: Functions without a name are called anonymous functions. They are often used as arguments to other functions or as immediately invoked function expressions (IIFE).

Code:

- Implementation of variables, data types, and typeof function

Ans:- Code:-

```
let enrollmentNo = 23162581026n;
let studentName = "Sujal Suthar";
let age = 19;
let isStudent = true;
let organization;
let score = null;
let uniqueId = Symbol("1005");

console.log("Enrollment No:", enrollmentNo);
console.log("Name:", studentName);
console.log("Age:", age);
console.log("Is Student:", isStudent);
console.log("Organization:", organization);
console.log("Score:", score);
console.log("Unique ID:", uniqueId);

console.log("Data Types:");
console.log("Enrollment No:", typeof enrollmentNo);
console.log("Name:", typeof studentName);
console.log("Age:", typeof age);
console.log("Is Student:", typeof isStudent);
console.log("Organization:", typeof organization);
console.log("Score:", typeof score);
console.log("Unique ID:", typeof uniqueId);
```

- Implementation of arrays and their methods

Ans:-
Code:-

```
let numbers = [15, 30, 25, 40, 50, 60, 75, 35, 45];

console.log("1. Join Method:");
console.log(" > Array as String:", numbers.join(" "), "\n");

numbers.push(5);
console.log("2. Push Method:");
console.log(" > Added '5' at the end:", numbers.join(" "), "\n");

numbers.pop();
console.log("3. Pop Method:");
console.log(" > Removed last element:", numbers.join(" "), "\n");

numbers.unshift(10);
console.log("4. Unshift Method:");
console.log(" > Added '10' at the beginning:", numbers.join(" "), "\n");

numbers.shift();
console.log("5. Shift Method:");
console.log(" > Removed first element:", numbers.join(" "), "\n");

let extraNumbers = [80, 85, 90];
let mergedArray = numbers.concat(extraNumbers);
console.log("6. Concatenation:");
console.log(" > Merged Array:", mergedArray.join(" "), "\n");

console.log("7. Reverse Method:");
console.log(" > Reversed Array:", numbers.reverse().join(" "));
```

- Implementation of objects and their methods.

Ans:- Code:-

```
let student = {
   "ID": 1026,
   "Name": "Sujal Suthar",
   "Batch": 44,
   "Semester": 4,
   "Enrollment No": 23162581026,
   "CGPA": { "Sem1": 6.5, "Sem2": 7.2 },
   "Email": "sujalsuthar@example.com",
   "Contact": 9876543210,
   "City": "Ahmedabad",
   "State": "Gujarat"
};

console.log(" > Display CGPA: ");
console.table(student.CGPA);

console.log(" > Display CGPA of Semester 2: " + student.CGPA.Sem2);

console.log(" > Object Keys:", Object.keys(student));
console.log(" > Object Values of CGPA:", Object.values(student.CGPA).join(" "));

console.log(" > Object Entries:");
console.table(Object.entries(student));

console.log(" > Iterating over Object Properties: ");
for (let key in student) {
   console.log(`${key}: ${student[key]}`);
}
```

- Implementation of different types of functions.

Ans:- Code:-

Normal Function:-

```
 console.log("Normal Function: ");

function performCalculation(a, b, operator) {
    switch (operator) {
        case "+": return a + b;
        case "-": return a - b;
        case "*": return a * b;
        case "/": return a / b;
        default: return "Invalid Operator";
    }
}
```

```
console.log(` > Addition (5 + 10): ${performCalculation(5, 10, "+")}`);
console.log(` > Subtraction (5 - 10): ${performCalculation(5, 10, "-")}`);
console.log(` > Multiplication (5 * 10): ${performCalculation(5, 10, "*")}`);
console.log(` > Division (5 / 10): ${performCalculation(5, 10, "/")}`);
```

Anonymous Function:-

```
console.log("Anonymous Function:");

let calculate = function (a, b, operator) {
   return operator === "+" ? `Sum: ${a + b}` :
          operator === "-" ? `Difference: ${a - b}` :
          operator === "*" ? `Product: ${a * b}` :
          operator === "/" ? `Quotient: ${a / b}` : "Invalid Operator";
};

console.log(" > " + calculate(20, 10, "+"));
console.log(" > " + calculate(20, 10, "-"));
console.log(" > " + calculate(20, 10, "*"));
console.log(" > " + calculate(20, 10, "/"));
```

Arrow Function:-

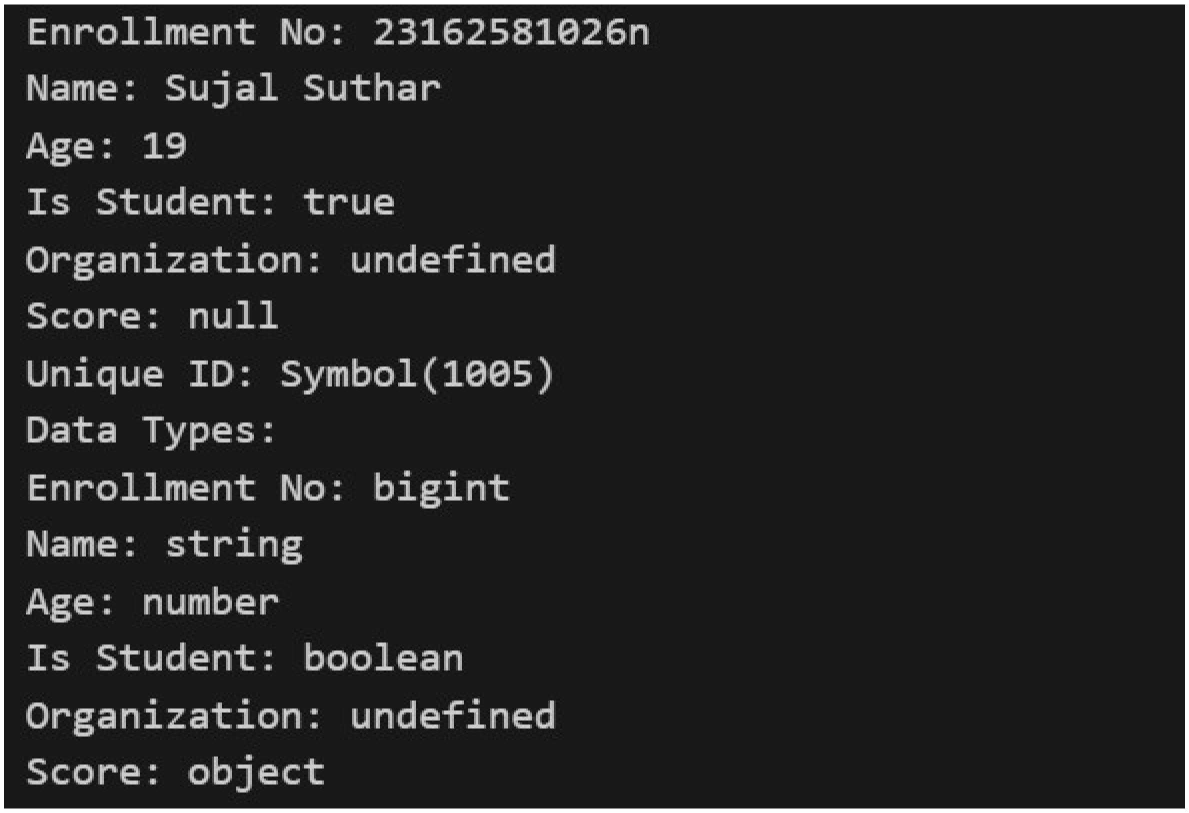
```
console.log("Arrow Function:");

let compute = (a, b, operator) => eval(`${a} ${operator} ${b}`);

console.log(` > Addition (30 + 20): ${compute(30, 20, "+")}`);
console.log(` > Subtraction (30 - 20): ${compute(30, 20, "-")}`);
console.log(` > Multiplication (30 * 20): ${compute(30, 20, "*")}`);
console.log(` > Division (30 / 20): ${compute(30, 20, "/")}`);
```

Output:

- Implementation of variables, data types, and typeof function

Ans:-

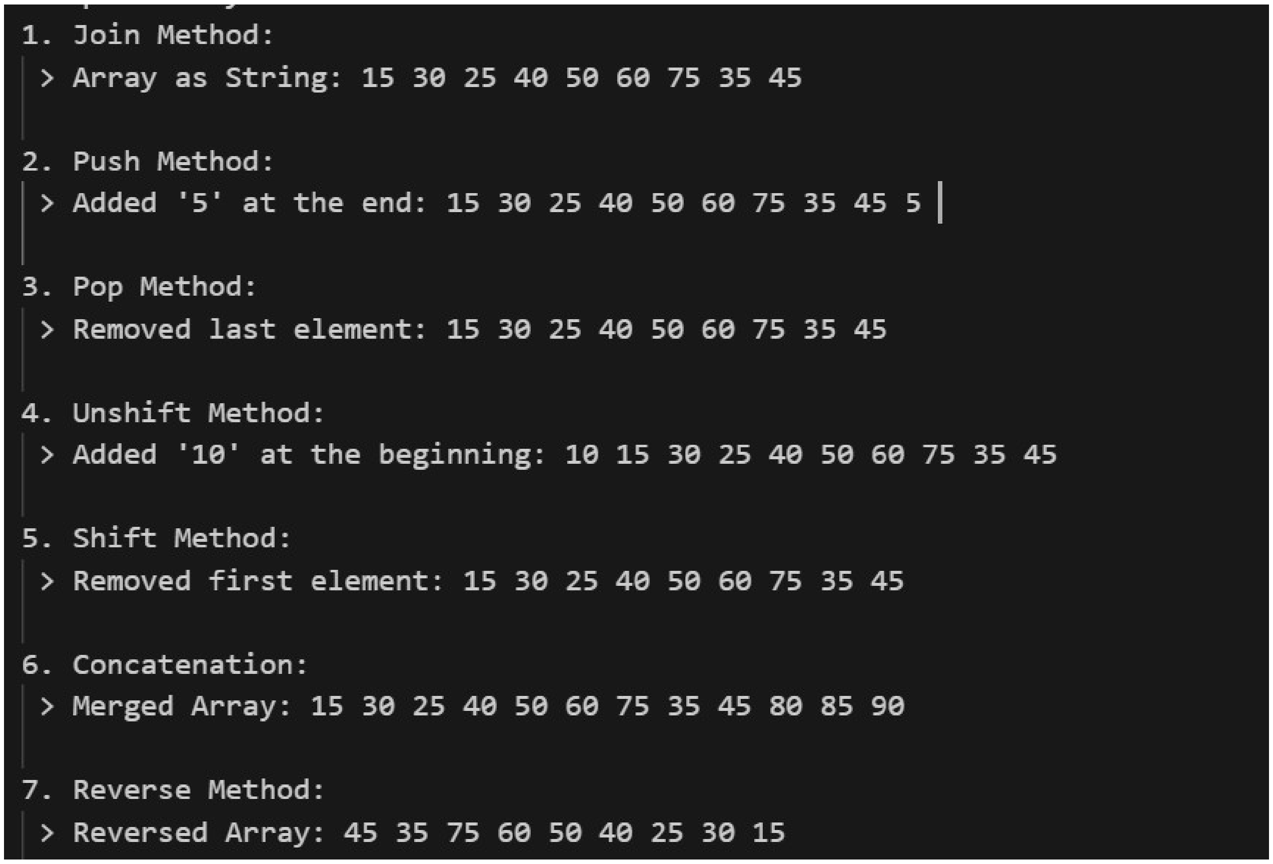
Output:-



```
Enrollment No: 23162581026n
Name: Sujal Suthar
Age: 19
Is Student: true
Organization: undefined
Score: null
Unique ID: Symbol(1005)
Data Types:
Enrollment No: bigint
Name: string
Age: number
Is Student: boolean
Organization: undefined
Score: object
```

- Implementation of arrays and their methods

Ans:-
Output:-

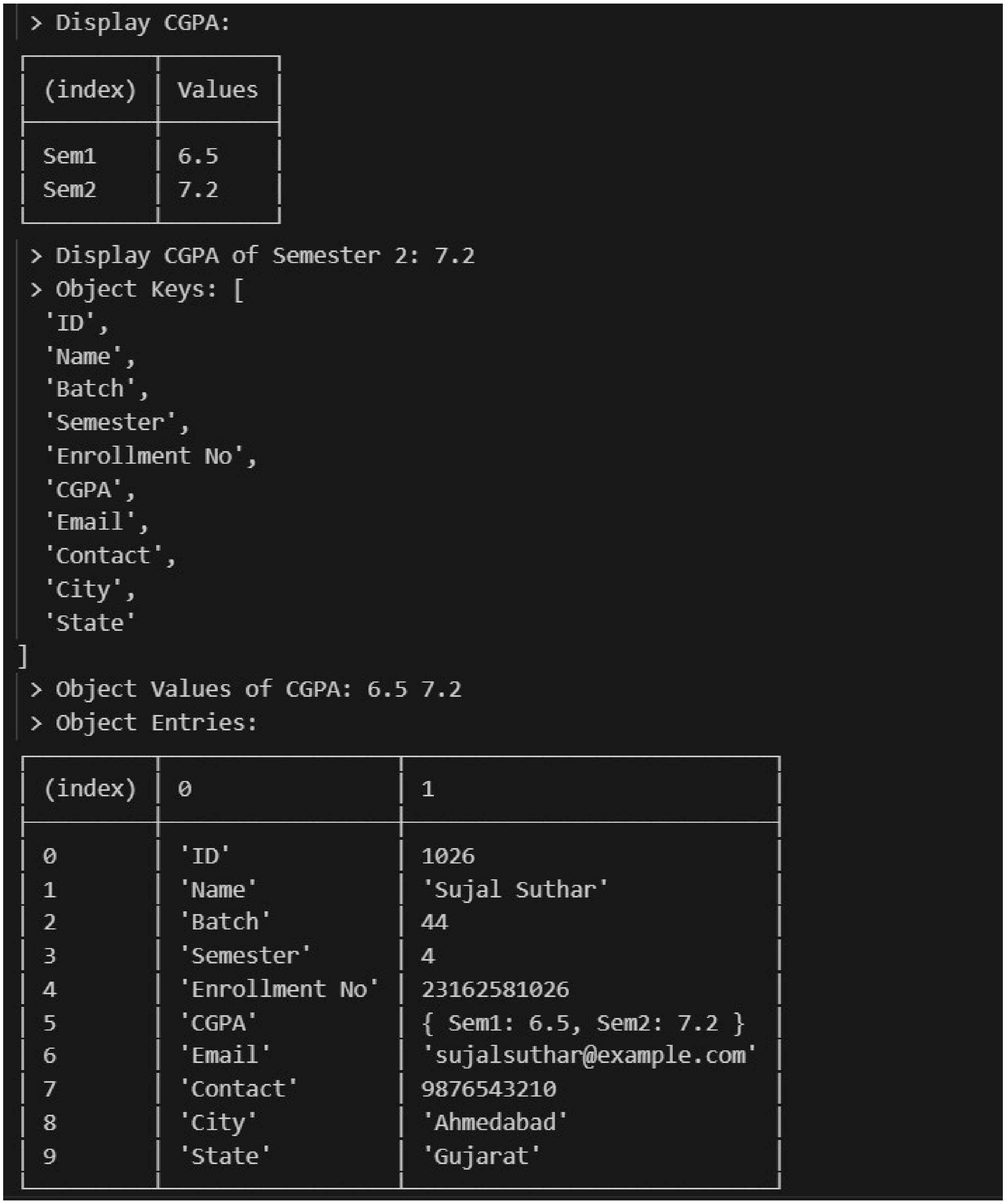


```
1. Join Method:
 > Array as String: 15 30 25 40 50 60 75 35 45

2. Push Method:
 > Added '5' at the end: 15 30 25 40 50 60 75 35 45 5

3. Pop Method:
 > Removed last element: 15 30 25 40 50 60 75 35 45

4. Unshift Method:
 > Added '10' at the beginning: 10 15 30 25 40 50 60 75 35 45

5. Shift Method:
 > Removed first element: 15 30 25 40 50 60 75 35 45

6. Concatenation:
 > Merged Array: 15 30 25 40 50 60 75 35 45 80 85 90

7. Reverse Method:
 > Reversed Array: 45 35 75 60 50 40 25 30 15
```

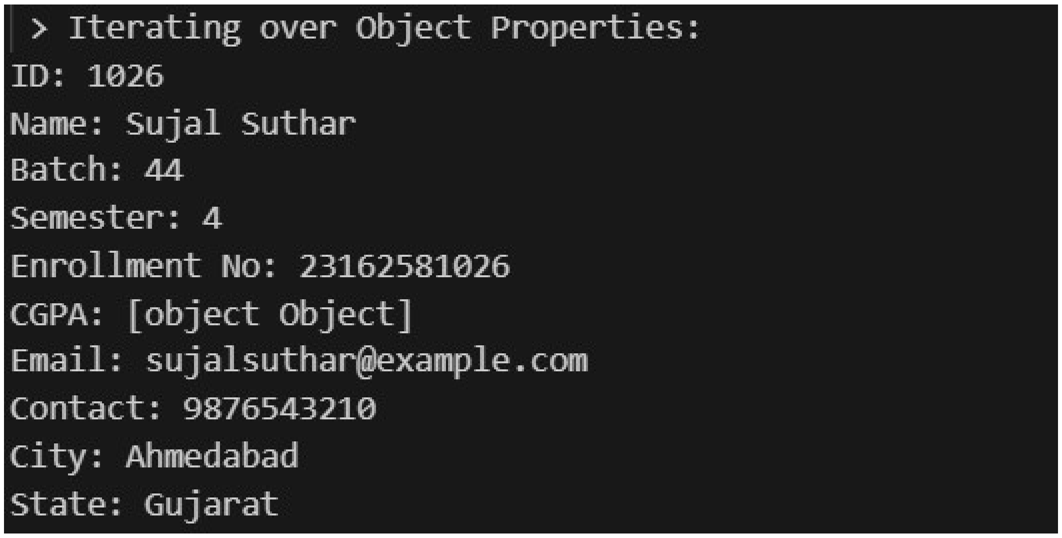
- Implementation of objects and their methods.

Ans:-

Output:-



```
> Display CGPA:
┌─────────┬────────┐
│ (index) │ Values │
├─────────┼────────┤
│ Sem1    │ 6.5    │
│ Sem2    │ 7.2    │
└─────────┴────────┘
> Display CGPA of Semester 2: 7.2
> Object Keys: [
  'ID',
  'Name',
  'Batch',
  'Semester',
  'Enrollment No',
  'CGPA',
  'Email',
  'Contact',
  'City',
  'State'
]
> Object Values of CGPA: 6.5 7.2
> Object Entries:
┌─────────┬─────────────────┬────────────────────────────┐
│ (index) │ 0               │ 1                          │
├─────────┼─────────────────┼────────────────────────────┤
│ 0       │ 'ID'            │ 1026                       │
│ 1       │ 'Name'          │ 'Sujal Suthar'             │
│ 2       │ 'Batch'         │ 44                         │
│ 3       │ 'Semester'      │ 4                          │
│ 4       │ 'Enrollment No' │ 23162581026                │
│ 5       │ 'CGPA'          │ { Sem1: 6.5, Sem2: 7.2 }   │
│ 6       │ 'Email'         │ 'sujalsuthar@example.com'  │
│ 7       │ 'Contact'       │ 9876543210                 │
│ 8       │ 'City'          │ 'Ahmedabad'                │
│ 9       │ 'State'         │ 'Gujarat'                  │
└─────────┴─────────────────┴────────────────────────────┘
```

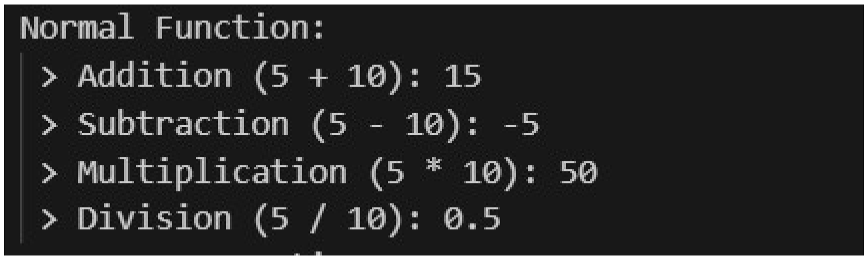


```
> Iterating over Object Properties:
ID: 1026
Name: Sujal Suthar
Batch: 44
Semester: 4
Enrollment No: 23162581026
CGPA: [object Object]
Email: sujalsuthar@example.com
Contact: 9876543210
City: Ahmedabad
State: Gujarat
```

- Implementation of different types of functions.

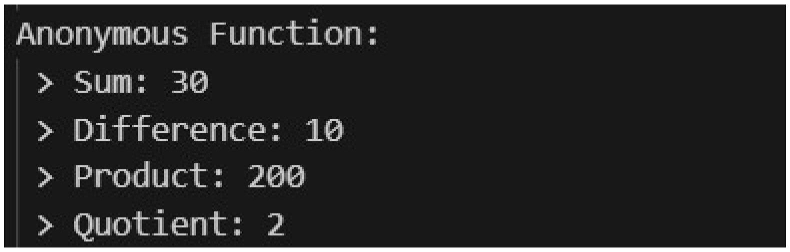
Ans:- Output:-

Normal Function:-



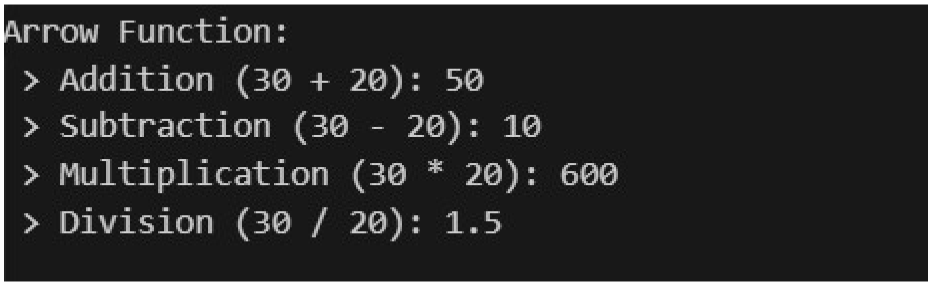
```
Normal Function:
 > Addition (5 + 10): 15
 > Subtraction (5 - 10): -5
 > Multiplication (5 * 10): 50
 > Division (5 / 10): 0.5
```

Anonymous Function:-



```
Anonymous Function:
 > Sum: 30
 > Difference: 10
 > Product: 200
 > Quotient: 2
```

Arrow Function:-



```
Arrow Function:
 > Addition (30 + 20): 50
 > Subtraction (30 - 20): 10
 > Multiplication (30 * 20): 600
 > Division (30 / 20): 1.5
```

Objective Achieved: Through this experiment, different data types and their categorization are understood under primitive and non-primitive data types. The arrays and objects and their methods are implemented. Creating a function is also understood and implemented.