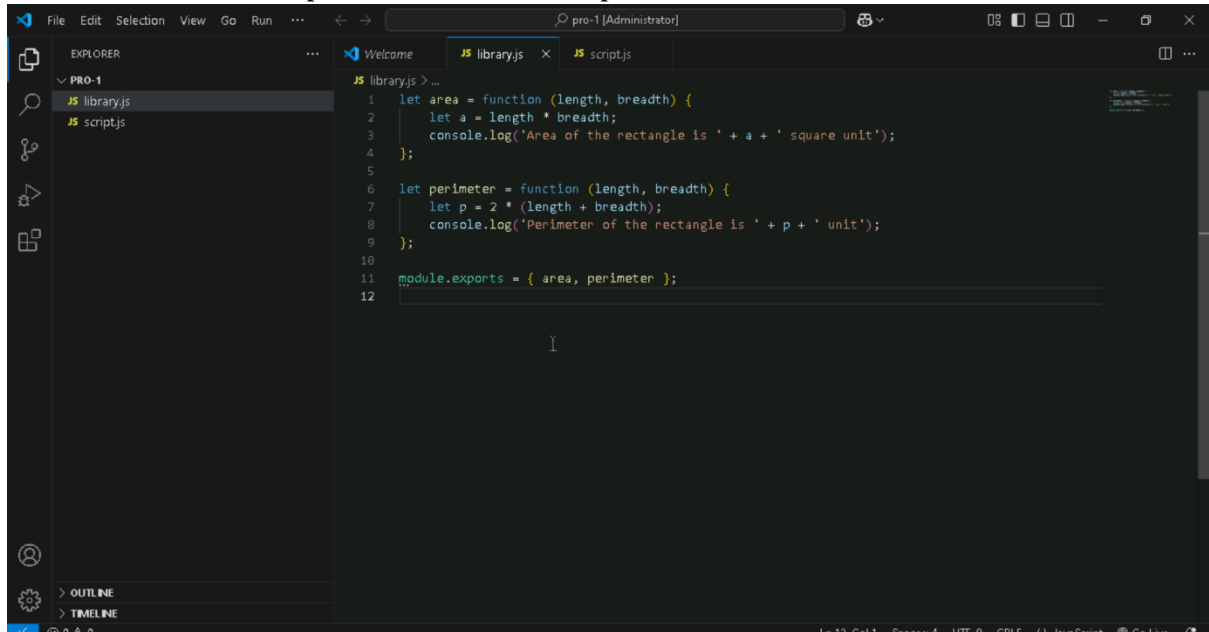


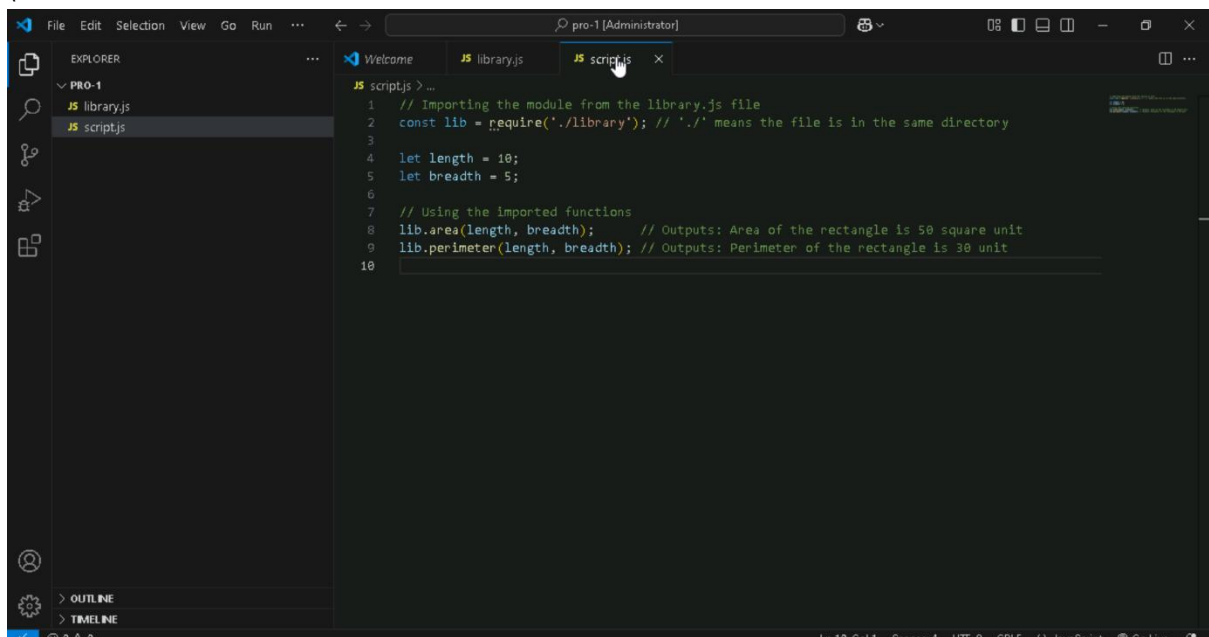
# JAVASCRIPT PROJECT

Create a module that exports a function and import it into another file.



This screenshot shows the Visual Studio Code editor with a project named 'pro-1'. The Explorer sidebar on the left shows two files: 'library.js' and 'script.js'. The main editor window displays the content of 'library.js', which defines two functions, 'area' and 'perimeter', and exports them using 'module.exports'.

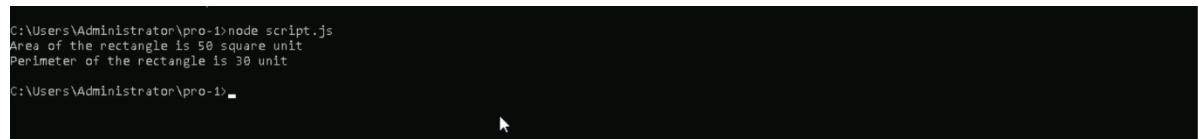
```
1 let area = function (length, breadth) {  
2   let a = length * breadth;  
3   console.log('Area of the rectangle is ' + a + ' square unit');  
4 }  
5  
6 let perimeter = function (length, breadth) {  
7   let p = 2 * (length + breadth);  
8   console.log('Perimeter of the rectangle is ' + p + ' unit');  
9 }  
10  
11 module.exports = { area, perimeter };  
12
```



This screenshot shows the Visual Studio Code editor with the same project 'pro-1'. The Explorer sidebar shows 'library.js' and 'script.js'. The main editor window displays the content of 'script.js', which imports the module from 'library.js' using 'require', sets variables for length and breadth, and then calls the imported 'area' and 'perimeter' functions.

```
1 // Importing the module from the library.js file  
2 const lib = require('./library'); // './' means the file is in the same directory  
3  
4 let length = 10;  
5 let breadth = 5;  
6  
7 // Using the imported functions  
8 lib.area(length, breadth); // Outputs: Area of the rectangle is 50 square unit  
9 lib.perimeter(length, breadth); // Outputs: Perimeter of the rectangle is 30 unit  
10
```

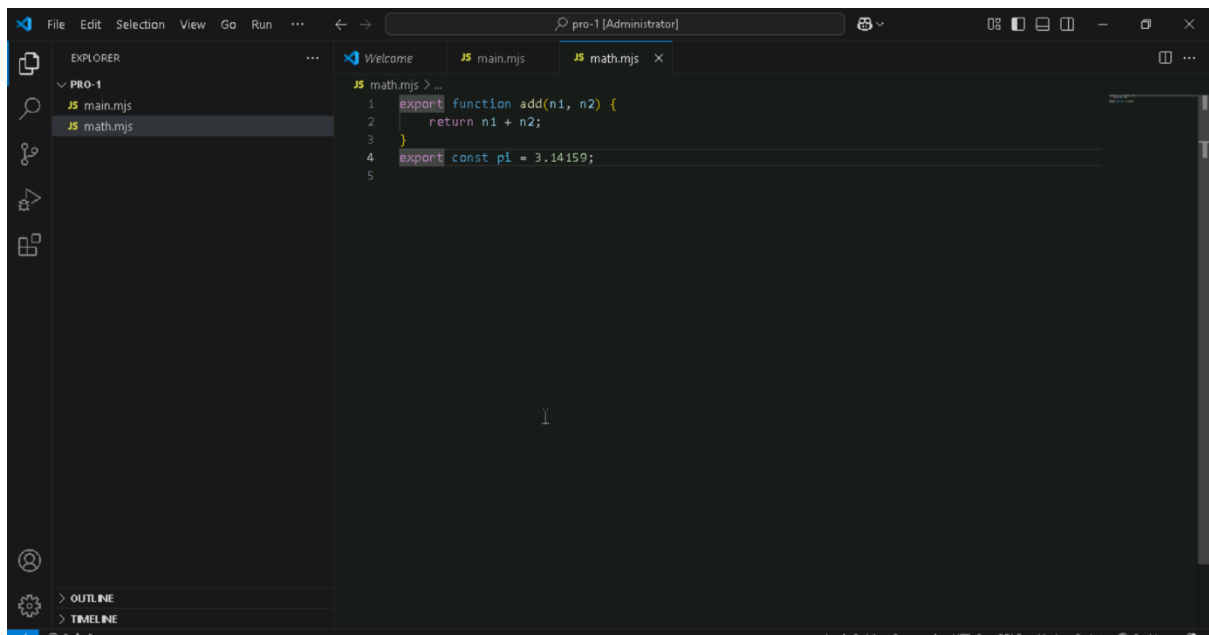
## OUTPUT



This screenshot shows a terminal window with the command 'node script.js' executed in the directory 'C:\Users\Administrator\pro-1'. The output of the program is displayed, showing the calculated area and perimeter of a rectangle with length 10 and breadth 5.

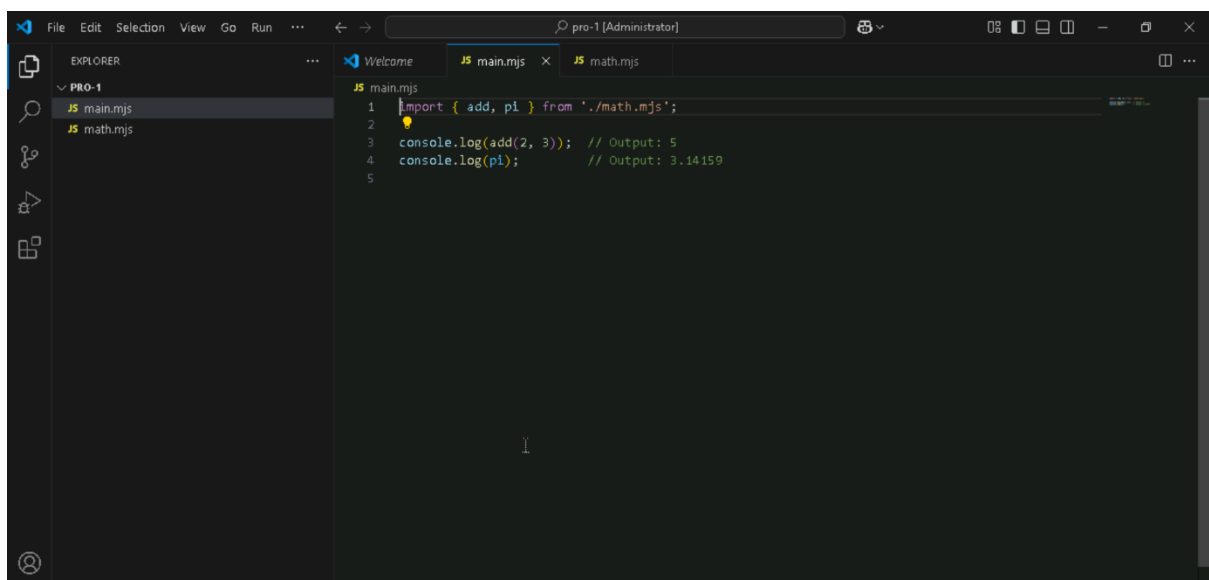
```
C:\Users\Administrator\pro-1>node script.js  
Area of the rectangle is 50 square unit  
Perimeter of the rectangle is 30 unit  
C:\Users\Administrator\pro-1>
```

Convert a JavaScript object to JSON and back using `JSON.stringify()` and `JSON.parse()`.



The screenshot shows the Visual Studio Code editor with a project named 'pro-1'. The Explorer sidebar on the left shows two files: 'main.mjs' and 'math.mjs'. The 'math.mjs' file is open in the editor, displaying the following code:

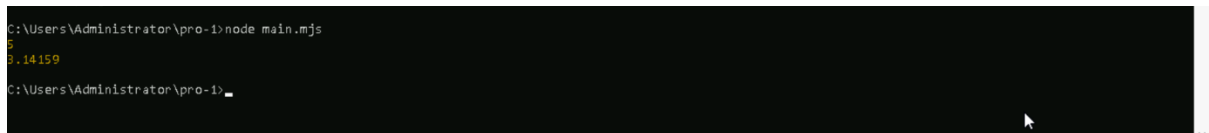
```
1 export function add(n1, n2) {  
2   return n1 + n2;  
3 }  
4 export const pi = 3.14159;  
5
```



The screenshot shows the Visual Studio Code editor with the 'main.mjs' file open. The code in the editor is as follows:

```
1 import { add, pi } from './math.mjs';  
2  
3 console.log(add(2, 3)); // Output: 5  
4 console.log(pi); // Output: 3.14159  
5
```

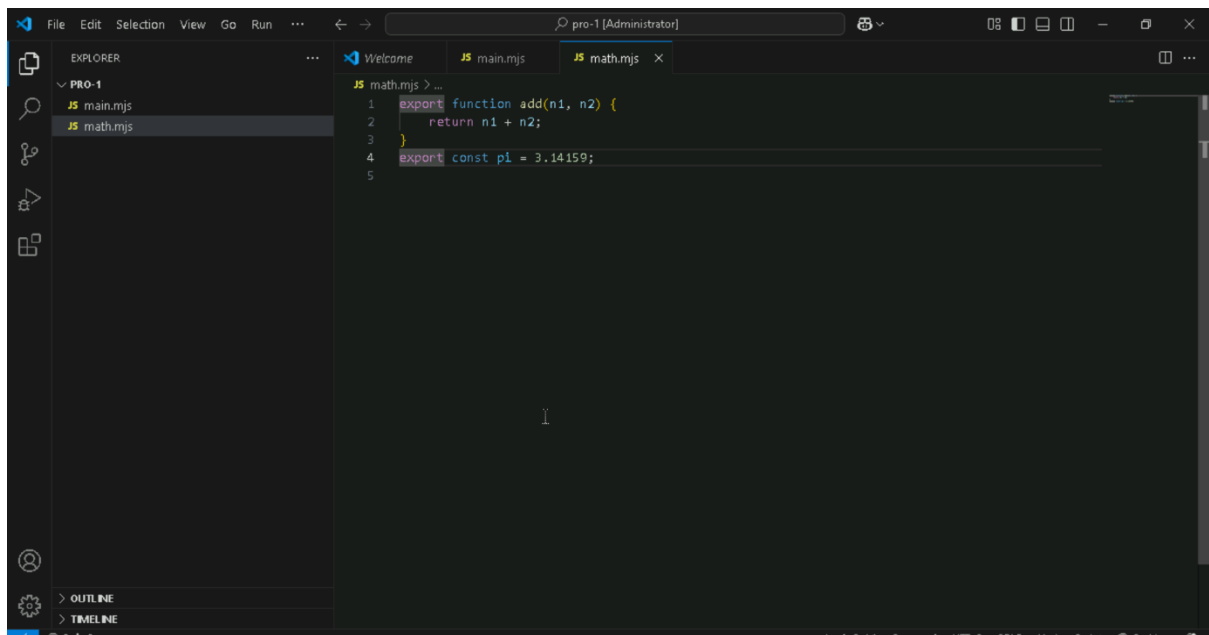
## OUTPUT



The screenshot shows a terminal window with the following output:

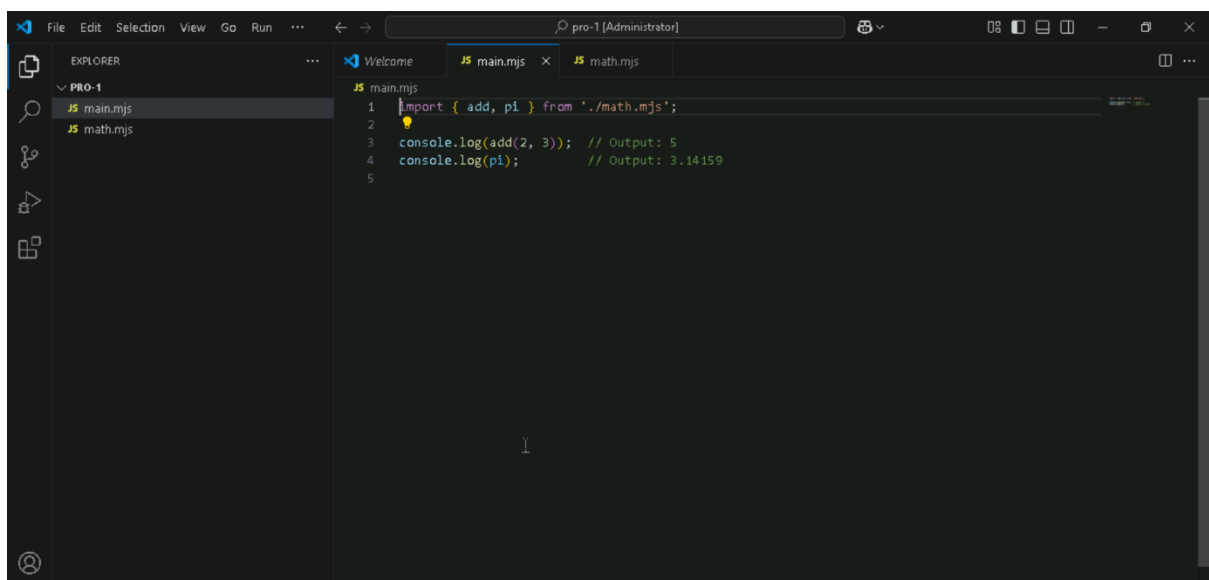
```
C:\Users\Administrator\pro-1>node main.mjs  
5  
3.14159  
C:\Users\Administrator\pro-1>
```

## Using require() Method (CommonJS)



The screenshot shows the Visual Studio Code editor with a project named 'pro-1'. The Explorer sidebar on the left shows two files: 'main.mjs' and 'math.mjs'. The 'math.mjs' file is open in the editor, displaying the following code:

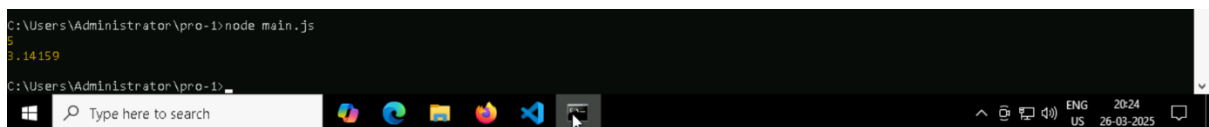
```
1 export function add(n1, n2) {  
2   return n1 + n2;  
3 }  
4 export const pi = 3.14159;  
5
```



The screenshot shows the Visual Studio Code editor with the 'main.mjs' file open. The code in the editor is as follows:

```
1 import { add, pi } from './math.mjs';  
2  
3 console.log(add(2, 3)); // Output: 5  
4 console.log(pi); // Output: 3.14159  
5
```

## OUTPUT



The screenshot shows a terminal window with the following command and output:

```
C:\Users\Administrator\pro-1>node main.js  
5  
3.14159  
C:\Users\Administrator\pro-1>
```

## Define a class Animal with properties and a method, and create instances



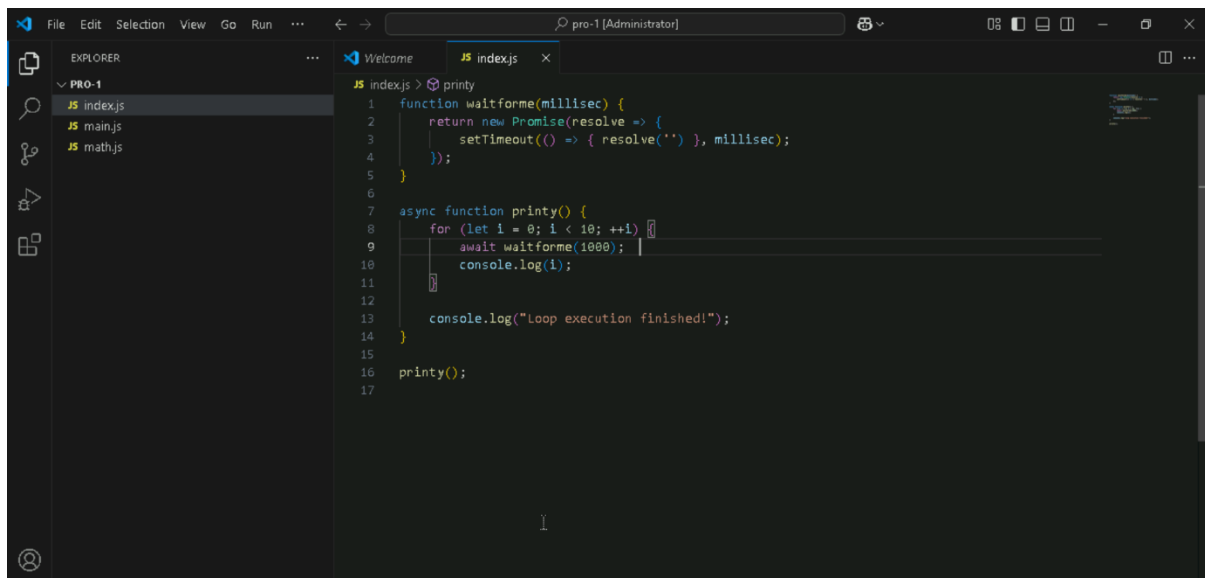
The screenshot shows a web-based JavaScript editor with a file named `index.js`. The code defines a class `Animal` with a constructor and a `makeSound` method. It then creates two instances, `cat` and `dog`, and calls their `makeSound` methods. The output on the right shows the results of these calls.

```
1 class Animal {
2   constructor(name, sound) {
3     this.name = name;
4     this.sound = sound;
5   }
6
7   makeSound() {
8     console.log(`${this.name} says ${this.sound}`);
9   }
10 }
11
12 const cat = new Animal('Cat', 'Meow');
13 const dog = new Animal('Dog', 'Woof');
14
15 cat.makeSound();
16 dog.makeSound();
17
```

Output:

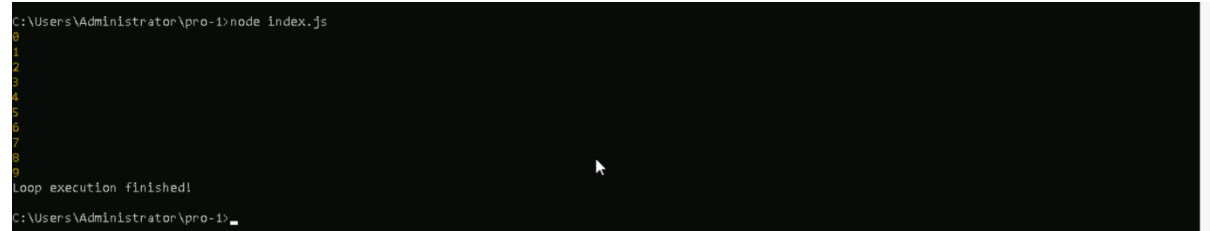
```
Cat says Meow
Dog says Woof
```

## Write a promise that resolves after a delay and convert it into an async function using async/await.



The screenshot shows Visual Studio Code with a file named `index.js`. The code defines a `waitforme` function that returns a promise that resolves after a specified delay. It then defines an async function `printy` that uses `await` to call `waitforme` in a loop. Finally, it calls `printy`.

```
1 function waitforme(millsec) {
2   return new Promise(resolve => {
3     setTimeout(() => { resolve('') }, millsec);
4   });
5 }
6
7 async function printy() {
8   for (let i = 0; i < 10; ++i) {
9     await waitforme(1000);
10    console.log(i);
11  }
12
13  console.log("Loop execution finished!");
14 }
15
16 printy();
17
```



The screenshot shows a terminal window with the command `node index.js` executed. The output shows the numbers 0 through 9, followed by the message "Loop execution finished!".

```
C:\Users\Administrator\pro-1>node index.js
0
1
2
3
4
5
6
7
8
9
Loop execution finished!
C:\Users\Administrator\pro-1>
```