Certainly! Here is the specific part of the code with detailed comments explaining each line:

```php

// Check if the 'loggedin' session variable is not set or is not true.

if (!isset($\_SESSION['loggedin']) || !$\_SESSION['loggedin']) {

// If the 'loggedin' session variable is not set or is false, redirect the user to 'login.php'.

header('Location: login.php');

// Terminate the script execution to ensure no further code is executed after the redirection.

exit;

}

1. \*\*`if (!isset($\_SESSION['loggedin']) || !$\_SESSION['loggedin'])`:\*\*

- This `if` statement checks two conditions to verify if the user is logged in.

* !isset($\_SESSION['loggedin'])`:\*\* This part checks if the `loggedin` session variable is not set.

`isset()` returns `true` if the variable exists and is not `null`. The `!` operator negates this, so it returns `true` if `$\_SESSION['loggedin']` is not set.

- \*\*`!$\_SESSION['loggedin']`:\*\* This part checks if the `loggedin` session variable is not `true`. The `!` operator negates the value of `$\_SESSION['loggedin']`, so it returns `true` if `$\_SESSION['loggedin']` is `false` or `0`.

2. \*\*`header('Location: login.php');`:\*\*

- If either of the conditions in the `if` statement is `true`, this line executes.

- The `header()` function sends a raw HTTP header to the browser. In this case, it sends a `Location` header to redirect the browser to `login.php`.

3. \*\*`exit;`:\*\*

- The `exit` function terminates the script execution immediately.

- This ensures that no further code is executed after the redirection, preventing unauthorized access to the rest of the page's content.

define('STUDENTS\_FILE', 'students.json');

Certainly! Here is the specific line of code with detailed comments explaining its purpose:

```php

// Define a constant named 'STUDENTS\_FILE' with the value 'students.json'.

// This constant will be used to store the path to the JSON file that contains student data.

define('STUDENTS\_FILE', 'students.json');

### Detailed Explanation

1. \*\*`define('STUDENTS\_FILE', 'students.json');`:\*\*

- **This line of code uses the `define()` function to create a constant.**

- \*\*`define()`\*\*: This function defines a constant, which is a name or an identifier for a simple value. **Once a constant is defined, it cannot be changed or undefined**.

- \*\*`**'STUDENTS\_FILE'**`\*\*: *This is the name of the constant. By convention, constant names are usually written in uppercase letters.*

- \*\*`'**students.json**'`\*\*: This is the value assigned to the constant `STUDENTS\_FILE`. **It represents the filename of the JSON file that will store the student data.**

- By defining this constant, we can use `STUDENTS\_FILE` throughout the script instead of hardcoding the filename `students.json` in multiple places. This makes the code more maintainable and easier to update if the filename ever changes.

Certainly! Here is the function with detailed comments explaining each part:

```php

function readStudents() {

// Check if the JSON file defined by STUDENTS\_FILE exists.

// If the file does not exist, return an empty array.

if (!file\_exists(STUDENTS\_FILE)) {

return [];

}

// Read the entire contents of the JSON file into a string.

$json = file\_get\_contents(STUDENTS\_FILE);

**// Decode the JSON string into a PHP array and return it**.

// The second parameter 'true' tells json\_decode to return an associative array instead of an object.

return json\_decode($json, true);

}

### Detailed Explanation

1. **\*\*`function readStudents() {`:\*\***

**- This line defines a function named `readStudents`. This function will read student data from a JSON file and return it as a PHP array.**

**2. \*\*`if (!file\_exists(STUDENTS\_FILE)) {`:\*\***

**- This line checks if the file specified by the constant `STUDENTS\_FILE` (which is `'students.json'`) exists using the `file\_exists()` function.**

- \*\*`file\_exists(STUDENTS\_FILE)`\*\*: This function returns `true` if the file exists and `false` otherwise.

- The `!` operator negates the result, so the condition is `true` if the file does not exist.

3. **\*\*`return [];`:\*\***

**- If the file does not exist, this line returns an empty array.**

- This is a safe default, ensuring that the function always returns an array, even if no student data is available.

4. \*\*`$json = file\_get\_contents(STUDENTS\_FILE);`:\*\*

- If the file exists, this line reads the entire contents of the file into a string using the `file\_get\_contents()` function.

- \*\*`file\_get\_contents(STUDENTS\_FILE)`\*\*: This function reads the file specified by `STUDENTS\_FILE` and returns its contents as a string.

5. \*\*`return json\_decode($json, true);`:\*\*

- This line decodes the JSON string into a PHP array using the `json\_decode()` function.

- \*\*`json\_decode($json, true)`\*\*: This function converts the JSON string `$json` into a PHP array. The second parameter `true` ensures that the JSON objects are converted to associative arrays rather than PHP objects.

- The decoded array is then returned by the function.

**### Summary**

* The `**readStudents**` function checks if the `students.json` file exists. If it does not, the function returns an empty array
* . **If the file exists, the function reads its contents**, **decodes the JSON data into a PHP array**, and returns this array.
* This provides a way to retrieve the student data stored in the JSON file in a structured format that can be easily used in the PHP script.