3.12 Server-Side Scripting using PHP

3.13 Introduction to PHP

PHP stands for Hypertext Preprocessor

It is a server-side scripting language used to create dynamic web pages

PHP is widely used in web development and can be integrated with HTML, CSS, and JavaScript

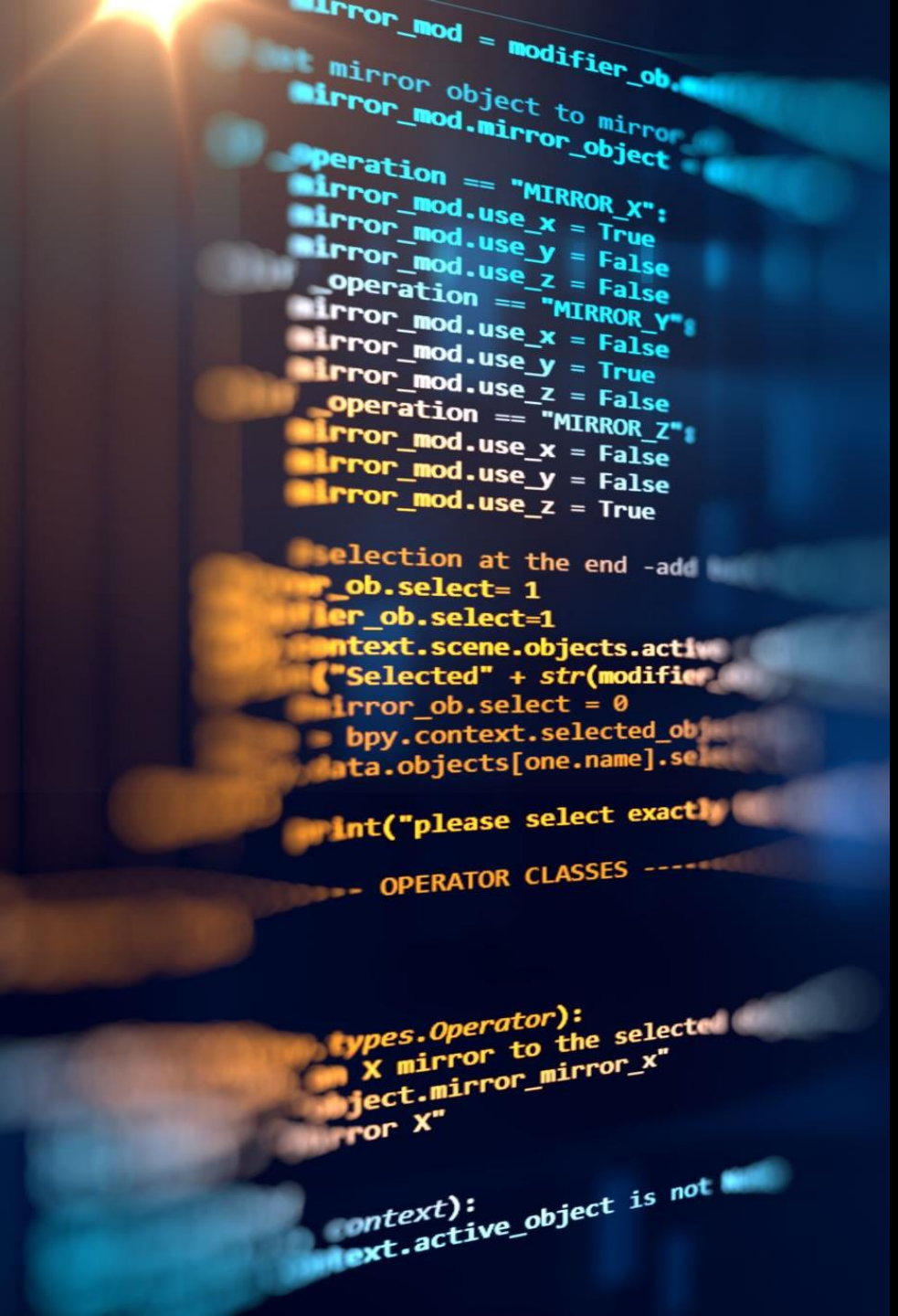
PHP code is executed on the server and the output is sent to the client's web browser

3.13 Hardware and Software Requirements

- PHP can run on various operating systems, including Windows, Mac, and Linux
- A web server is required to run PHP, such as Apache or Nginx
- PHP also requires a database system, such as MySQL or PostgreSQL
- A code editor, such as Sublime Text, Visual Studio Code, or PHPStorm, can be used to write and edit PHP code
- A web browser is used to access the code via a URL

3.14 Object Oriented Programming in PHP

- PHP supports object-oriented programming (OOP)
- OOP allows you to model real-world objects in your code
- The main concepts in OOP are classes, objects, properties, and methods
- Classes define the blueprint for an object, and an object is an instance of a class
- Properties are the variables that store the state of an object, and methods are the functions that define the behavior of an object
- OOP allows for code reusability, encapsulation, and inheritance
- PHP provides built-in support for OOP, including the ability to define classes, create objects, and inherit properties and methods from parent classes.
- When using OOP in PHP, you will use keywords like class, extends, implements, public, protected, private.
- It's important to understand the basics of OOP and how to use classes, objects, properties, and methods in order to write modular, maintainable, and efficient code.



Setting up a PHP Development Environment

- PHP can be installed on a local machine or on a web server
- A local development environment can be set up using a web server software such as XAMPP or WAMP
- A code editor such as Sublime Text, Visual Studio Code, or PHPStorm can be used to write and edit PHP code
- To run PHP code, a web browser is used to access the code via a URL

3.15 Basic PHP Syntax

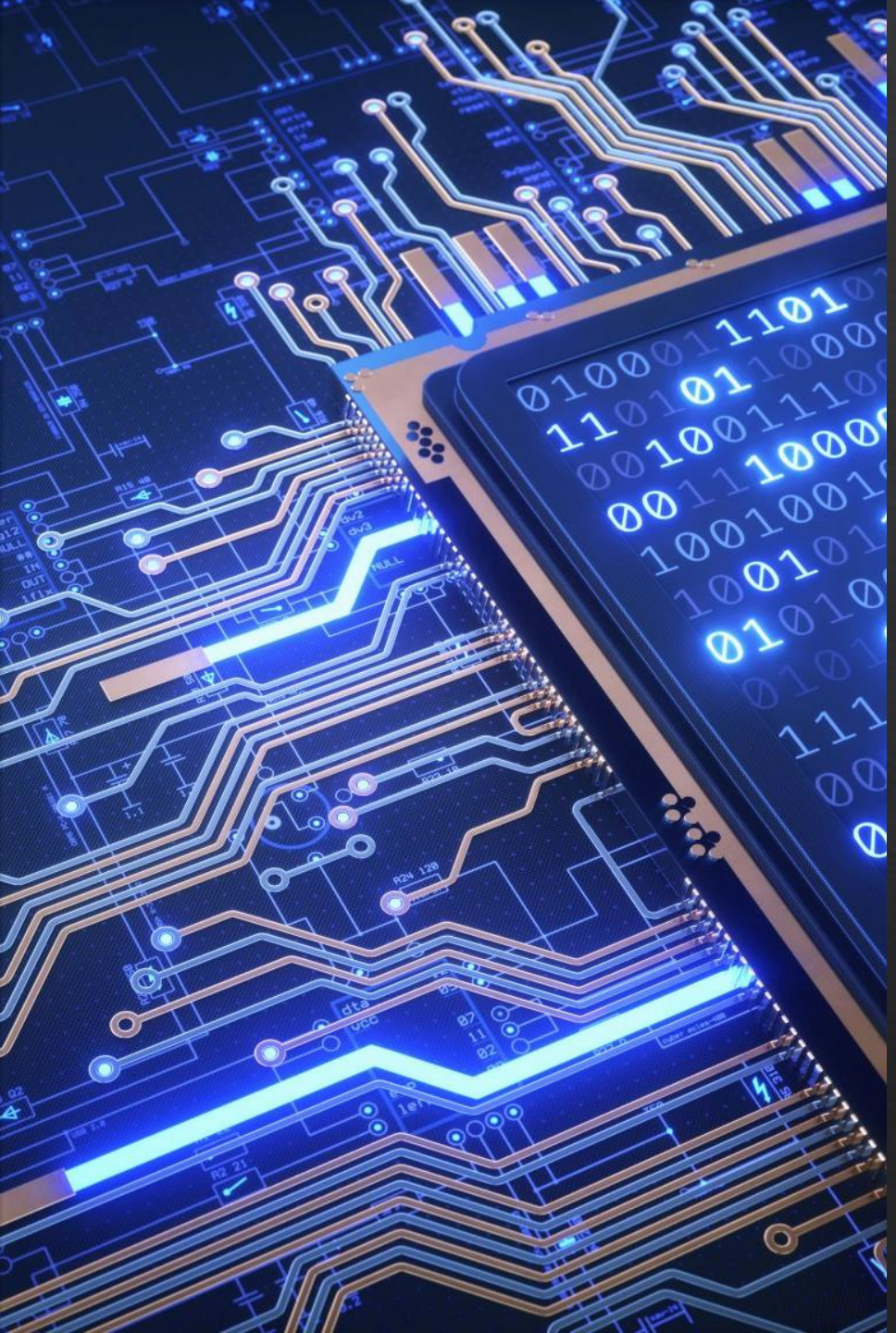
- PHP code is enclosed within PHP tags <?php ?>
- Variables start with a dollar sign ($), and are case-sensitive
- Comments can be added in PHP using // or /* */
- PHP supports a variety of control structures, including if/else, switch, and loops
- PHP also supports arrays and functions

3.16 PHP Data Types

- PHP supports a variety of data types, including:
  - String: a sequence of characters, enclosed in single or double quotes
  - Integer: a whole number, without a decimal point
  - Float: a number with a decimal point
  - Boolean: a true or false value
  - Array: a collection of values, stored in a single variable
  - Object: an instance of a class, which can have properties and methods
- PHP automatically determines the data type of a variable based on its value
- The data type can be explicitly set by using type casting
- PHP also supports NULL which means the variable has no value assigned

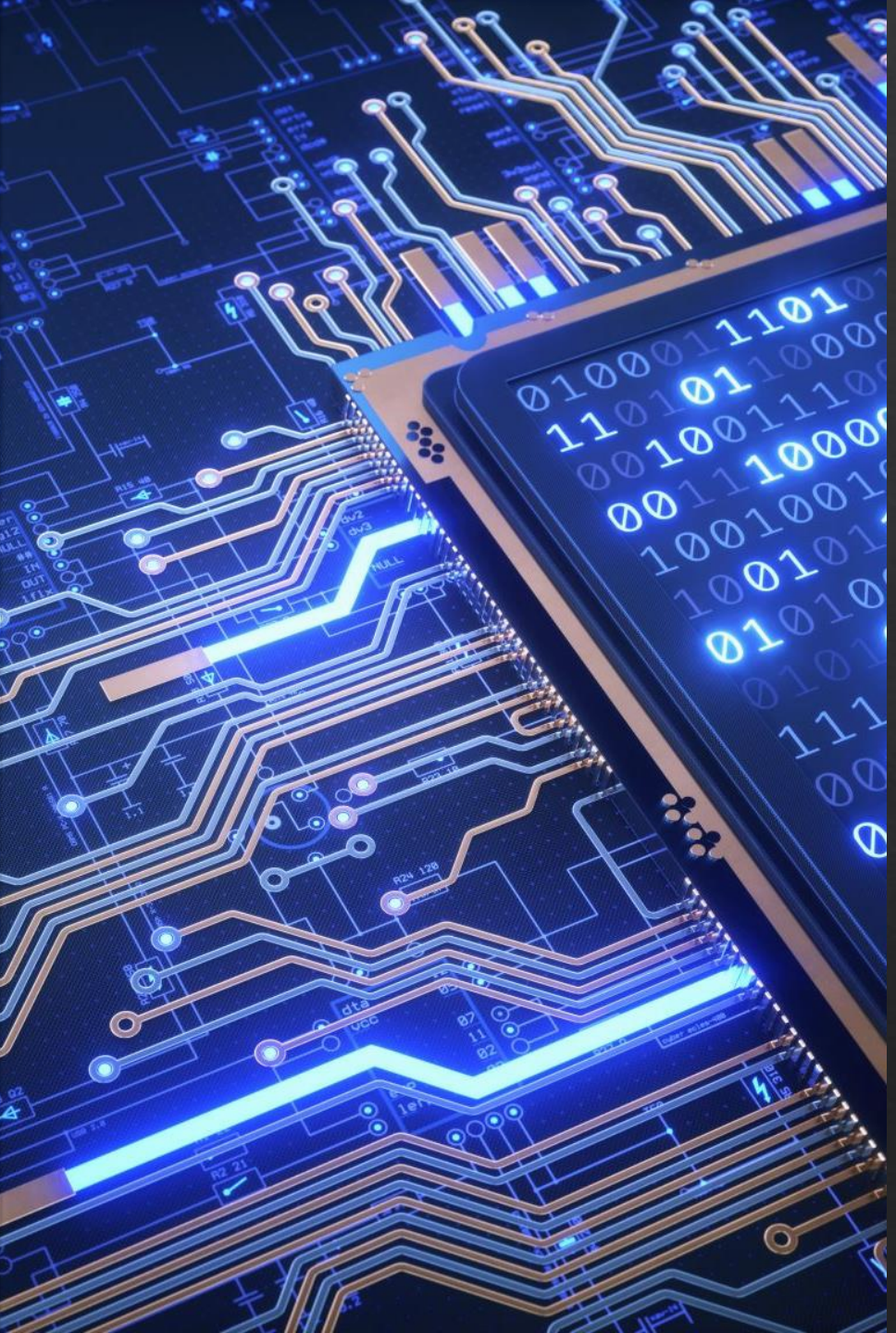
3.16 More specific PHP Data Types

- PHP also has additional data types that are used less frequently such as :
  - Resource: a special variable that holds a reference to an external resource
  - Callback: a variable that holds a reference to a function
  - Mixed: a variable that can contain any type of data
  - Number: a variable that can be either an integer or a float
  - Array or ArrayObject: an advanced data structure that can store multiple values in a single variable
- these data types are used less frequently but can be very powerful when used correctly



3.17 Basic Programming in PHP

- Basic programming in PHP includes a variety of control structures, functions and arrays that allow you to organize and manipulate your code.
- Control Structures:
  - Conditional statements: if, else, elseif
  - Loops: for, while, do-while
- Functions:
  - User defined functions
  - Built-in functions
- Arrays:
  - Indexed arrays
  - Associative arrays
  - Multidimensional arrays



3.17 Basic Programming in PHP

- Here is an example of a basic program that uses a conditional statement, a loop and a function in PHP:

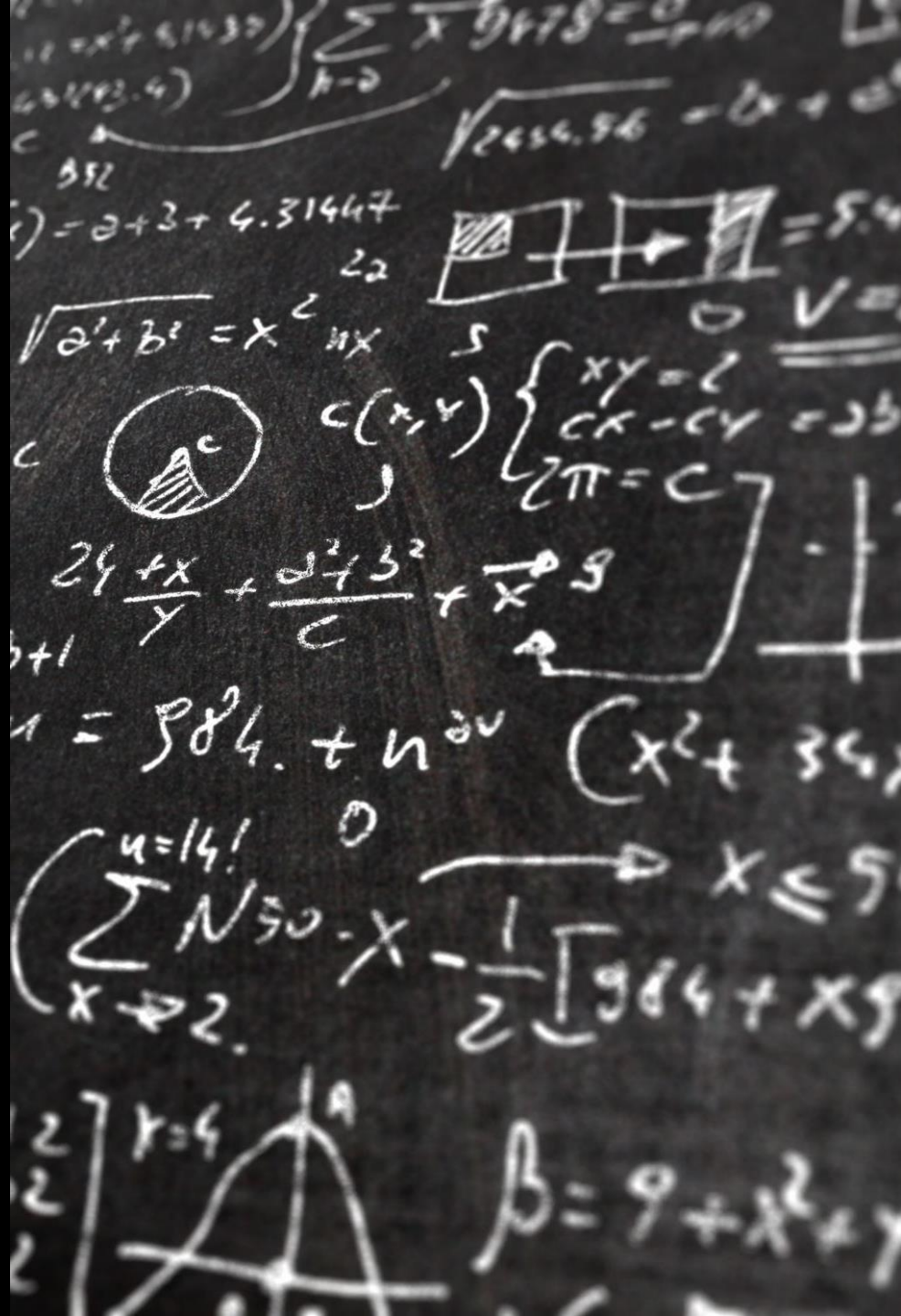
php Copy code

```php
<?php
$x = 5;

if ($x > 0) {
    echo "x is positive";
}

for ($i = 0; $i < $x; $i++) {
    echo $i;
}

function multiply($a, $b) {
    return $a * $b;
}

echo multiply(3, 4);
?>
```

- This slide provides a glimpse of basic programming in PHP, and it's not limited to this, you can use many more advanced features and functions of PHP to make your program sophisticated.

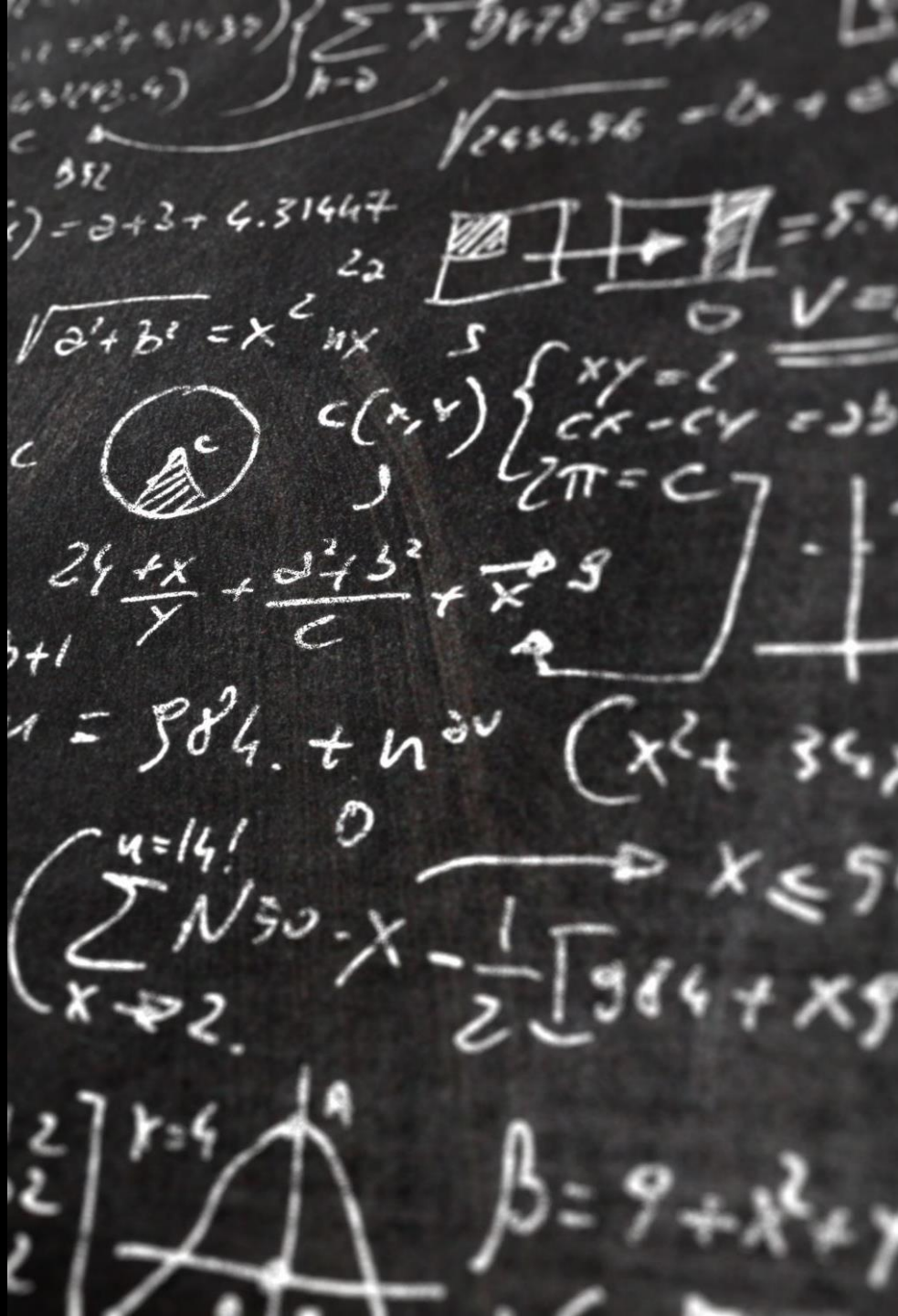
3.18 Operators in PHP

- Operators are used to perform operations on variables and values
- PHP supports various types of operators, including:
  - Arithmetic operators (e.g. +, -, *, /)
  - Comparison operators (e.g. ==, !=, >, <)
  - Logical operators (e.g. &&, ||, !)
  - Assignment operators (e.g. =, +=, -=)
- Operators can be used in expressions and conditions to control the flow of the program
- Order of precedence can be controlled with parenthesis
- It's also possible to chain the assignment operators for example $a = $b = $c = 0;

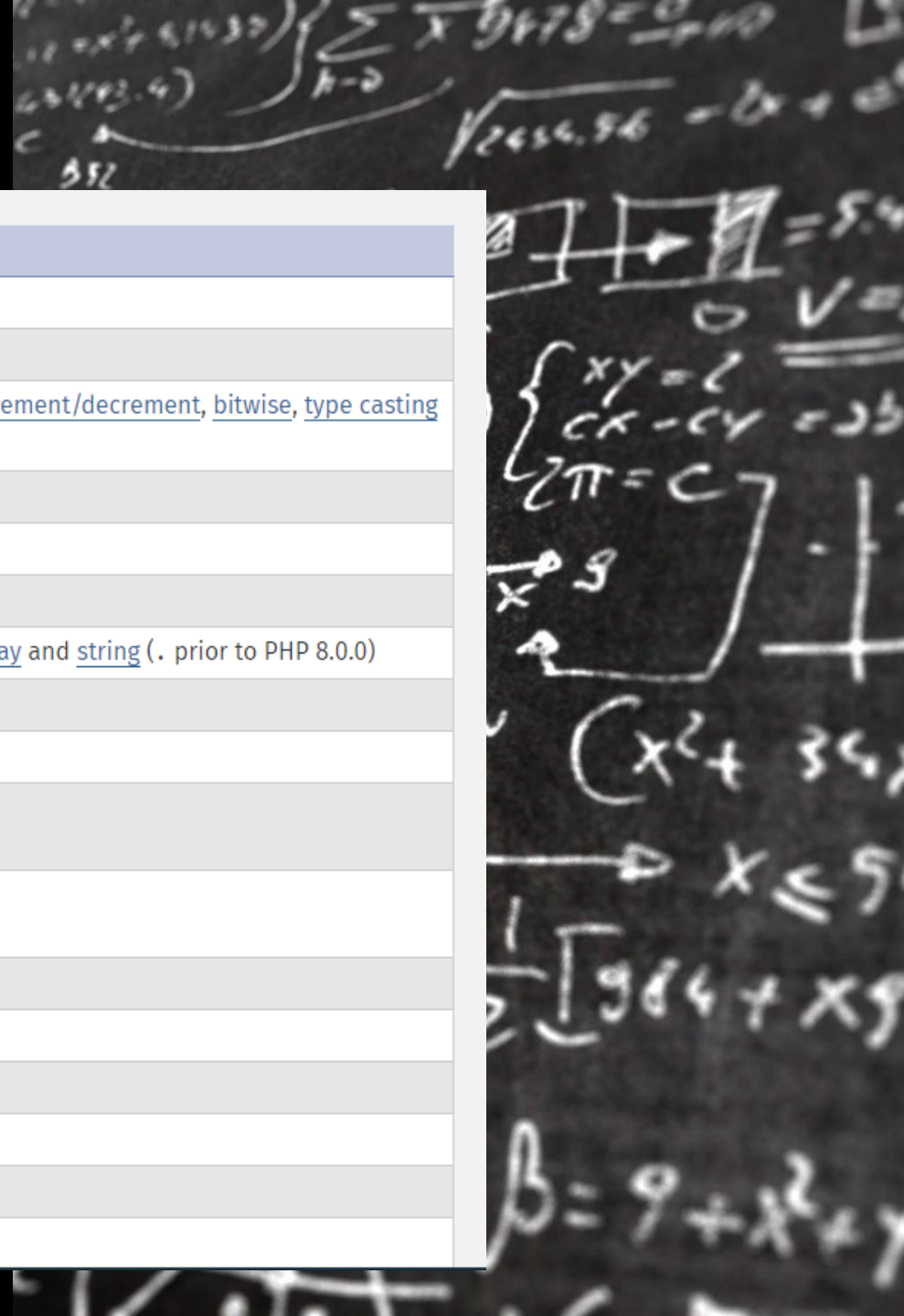


3.18 Operator Precedence in PHP

- Operator precedence determines the order in which operations are performed in an expression.
- In general, operators with higher precedence are executed before operators with lower precedence
- For example, multiplication and division have higher precedence than addition and subtraction
- Parentheses can be used to change the order of precedence and explicitly specify the order in which operations should be performed
- If there are multiple operators with the same precedence, they are evaluated left-to-right.



3.18 Operator Precedence in PHP

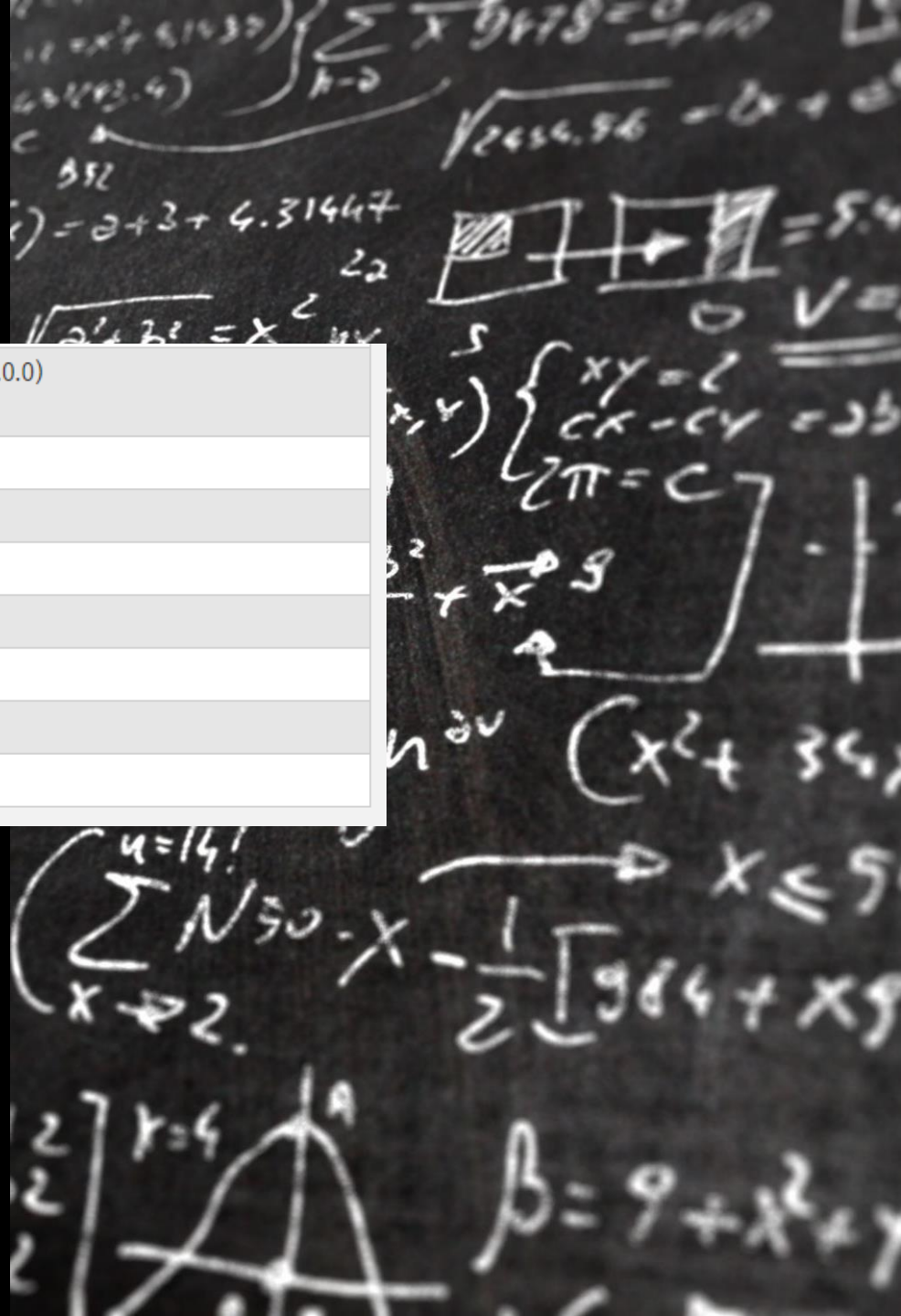


Operator Precedence

Associativity	Operators	Additional Information
(n/a)	`clone new`	clone and new
right	`**`	arithmetic
(n/a)	`+ - ++ -- ~ (int) (float) (string) (array) (object) (bool) @`	arithmetic (unary + and -), increment/decrement, bitwise, type casting and error control
left	`instanceof`	type
(n/a)	`!`	logical
left	`* / %`	arithmetic
left	`+ - .`	arithmetic (binary + and -), array and string (`.` prior to PHP 8.0.0)
left	`<< >>`	bitwise
left	`.`	string (as of PHP 8.0.0)
non-associative	`< <= > >=`	comparison
non-associative	`== != === !== <> <=>`	comparison
left	`&`	bitwise and references
left	`^`	bitwise
left	`|`	bitwise
left	`&&`	logical
left	`||`	logical
right	`??`	null coalescing

3.18 Operator Precedence in PHP

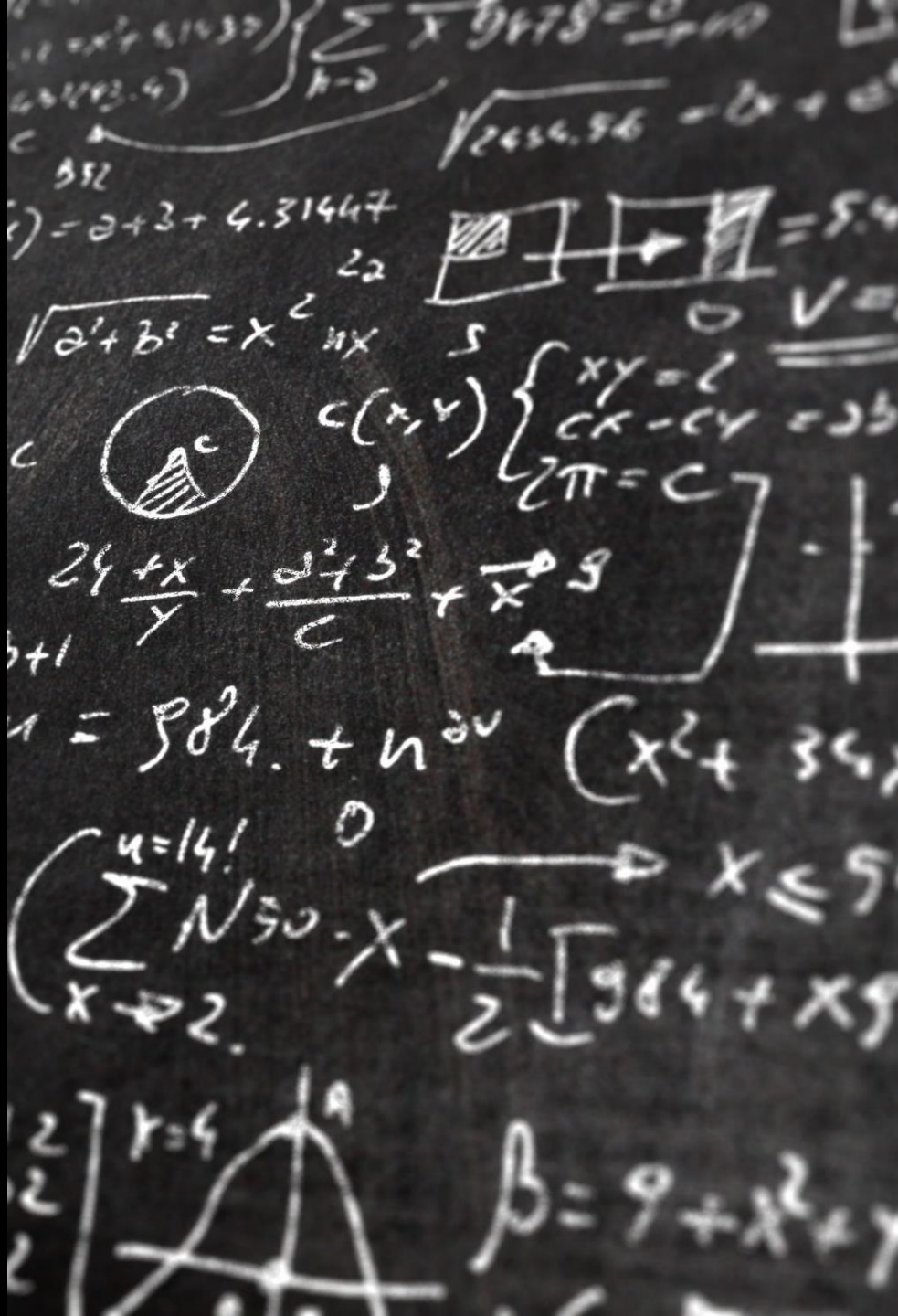
non-associative	`? :`	ternary (left-associative prior to PHP 8.0.0)
right	`= += -= *= **= /= .= %= &= |= ^= <<= >>= ??=`	assignment
(n/a)	`yield from`	yield from
(n/a)	`yield`	yield
(n/a)	`print`	print
left	`and`	logical
left	`xor`	logical
left	`or`	logical



3.18 Operator Precedence Examples:

1. In the expression 4 + 5 * 2, the multiplication (*) has a higher precedence than the addition (+), so the multiplication is done first.
2. In the expression (4 + 5) * 2, the parentheses change the order of operations, so the addition is done first.

- It's important to keep the operator precedence in mind when writing complex expressions, to ensure that the operations are performed in the order you expect.



3.19 Variables Manipulation

- PHP allows you to manipulate variables in a variety of ways, including:
  - Assign values to variables using the assignment operator (=)
  - Concatenate strings using the concatenation operator (.)
  - Manipulate arrays using array functions such as sort(), count(), and implode()
  - Use type casting to explicitly change the data type of a variable
  - Use the global keyword to access global variables within a function
  - Use the reference operator (&) to pass variables by reference
- It's important to understand how to manipulate variables in order to effectively use them in your code.

3.20 Database connectivity

- PHP can be used to connect to a wide variety of databases, including MySQL, MariaDB, PostgreSQL, SQLite, and Oracle
- PHP provides built-in support for connecting to databases through the use of extensions such as PDO (PHP Data Objects) and MySQLi
- PDO is a database-agnostic extension that allows you to connect to multiple types of databases using a single API
- MySQLi is a specific extension for working with MySQL databases
- Both PDO and MySQLi provide methods for executing SQL statements, binding parameters, and fetching results
- It's important to use prepared statements and parameter binding to protect against SQL injection attacks
- Additionally, it's good practice to use an ORM (Object-Relational Mapping) library to abstract the database operations and make the code more readable and maintainable

3.20 Database connectivity

- Examples of ORM library in PHP: Doctrine, Eloquent, and RedBeanPHP
- These libraries provide a simple, object-oriented API for working with databases, making it easy to perform common tasks such as inserting, updating, and deleting records, as well as querying and retrieving data
- ORM libraries also help to minimize the amount of SQL code you have to write, and they can handle tasks such as connecting to the database, creating tables, and defining relationships between tables
- ORM libraries are widely used in PHP web development to make the interaction with databases simple and secure
- They can also help to improve the performance of your application by caching frequently used data and reducing the number of database queries needed.

3.21 Connecting server-side script to database

- To connect to a database from PHP, you will need to use the appropriate database extension (e.g. PDO or MySQLi) and provide the necessary connection details such as the hostname, username, password, and database name
- Once connected, you can use the extension's methods to execute SQL statements, bind parameters, and fetch results
- It's important to use prepared statements and parameter binding to protect against SQL injection attacks
- It's also a good practice to use an ORM library to abstract the database operations and make the code more readable and maintainable
- An example of connecting to a MySQL database using PDO:

3.21 Connecting server-side script to database

php

Copy code

```php
$dsn = 'mysql:host=localhost;dbname=mydatabase';
$user = 'myusername';
$password = 'mypassword';
$pdo = new PDO($dsn, $user, $password);
```

3.22 Making SQL queries in PHP

- Once connected to the database, you can execute SQL statements using the appropriate methods of the database extension (e.g. PDO or MySQLi)
- For example, to execute a SELECT statement using PDO, you can use the query() method:

```php
$stmt = $pdo->query('SELECT * FROM mytable');
```

- To execute an INSERT statement using PDO, you can use the prepare() and execute() methods:

```php
$stmt = $pdo->prepare('INSERT INTO mytable (name, email) VALUES (?, ?)');
$stmt->execute(['John Doe', 'johndoe@example.com']);
```

3.22 Making SQL queries in PHP

- It's important to use prepared statements and parameter binding to protect against SQL injection attacks
- ORM libraries provide an even more simple and secure way to interact with databases.

3.23 Fetching data sets

- To retrieve data from a database, you can use the SELECT statement.
- SELECT statement is used to select data from a database.
- SELECT statement is used to select specific columns and rows from a table in a database
- SELECT statement is the most common statement in SQL
- Example of a SELECT statement:

```php
SELECT column1, column2, column3 FROM table_name;
```

3.23 Fetching data sets

- The WHERE clause is used to filter the results of a SELECT, UPDATE, or DELETE statement
- The WHERE clause is used to extract only those records that fulfill a specified condition
- It is used to filter the results of a SELECT, UPDATE, or DELETE statement
- Example of using WHERE clause:

```php
SELECT column1, column2, column3 FROM table_name WHERE column1 = value;
```

3.23 Fetching data sets

- The WHERE clause is used to filter the results of a SELECT, UPDATE, or DELETE statement
- The WHERE clause is used to extract only those records that fulfill a specified condition
- It is used to filter the results of a SELECT, UPDATE, or DELETE statement
- Example of using WHERE clause:

```php
SELECT column1, column2, column3 FROM table_name WHERE column1 = value;
```

3.24 Creating SQL database with server-side scripting using PHP

- Creating a SQL database with server-side scripting using PHP is a common task when building dynamic web applications. PHP provides several ways to interact with a SQL database, including the MySQLi and PDO extensions.
- To create a SQL database using PHP, you need to:

1. Connect to a MySQL server using the MySQLi or PDO extension.
2. Send a SQL query to create the database.
3. Close the connection.

3.24 Creating SQL database with server-side scripting using PHP

Here is an example of creating a SQL database using the MySQLi extension:

It's important to note that creating a SQL database using PHP can be executed only by a user with sufficient privileges.

php

Copy code

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```