Learning PHP involves mastering a variety of concepts, from basic syntax to advanced topics like object-oriented programming and frameworks. Here's a breakdown of key points in PHP from basic to advanced:

1. **Basic Syntax:**
   - Variables and Data Types: Scalars (integers, floats, strings, booleans), Arrays, Objects
   - Operators: Arithmetic, Comparison, Logical, Assignment
   - Control Structures: if...else, switch, loops (for, while, foreach)
   - Functions: Built-in functions, User-defined functions

2. **Advanced Syntax:**
   - Error Handling: try...catch, error_reporting
   - File Handling: Reading from and writing to files, file system functions
   - Regular Expressions: Pattern matching using preg_match(), preg_replace(), etc.
   - Super Global Variables: $_GET, $_POST, $_SESSION, $_COOKIE, $_FILES

3. **Object-Oriented Programming (OOP):**
   - Classes and Objects: Properties, Methods, Constructors, Destructors
   - Inheritance, Encapsulation, Polymorphism
   - Interfaces, Abstract Classes
   - Namespaces

4. **Database Interaction:**
   - MySQLi and PDO: Connecting to databases, executing queries, fetching results
   - Prepared Statements: Preventing SQL injection attacks
   - Database Design: Normalization, Indexing, Relationships

5. **Web Development:**
   - HTTP Basics: GET and POST requests, response codes
   - Sessions and Cookies: Managing user sessions and data persistence
   - Form Handling: Processing form submissions, form validation
   - Security Measures: Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), Data Validation and Sanitization

6. **Advanced Topics:**
   - PHP Extensions: Utilizing extensions like cURL, GD (for image manipulation), etc.
   - Caching Techniques: Using APC, Memcached, or Redis for performance optimization
   - Web Services: Consuming and creating APIs (RESTful, SOAP)
   - Performance Optimization: Code profiling, caching strategies, opcode caching
   - Design Patterns: MVC, Singleton, Factory, Dependency Injection

7. **Frameworks and Libraries:**
   - Laravel, Symfony, CodeIgniter, Zend Framework: Full-stack frameworks for building robust web applications
   - Composer: Dependency manager for PHP
   - PHPUnit: Testing framework for unit testing PHP code
   - Twig, Smarty: Template engines for separating presentation from business logic

8. **Deployment and Hosting:**
   - Setting up PHP Environment: Apache, Nginx, PHP-FPM
   - Version Control: Git, SVN
   - Continuous Integration/Continuous Deployment (CI/CD) pipelines
   - Cloud Hosting: AWS, Google Cloud, Azure
   - Containerization: Docker, Kubernetes

Mastering these topics will give you a comprehensive understanding of PHP, allowing you to build anything from simple scripts to complex web applications. Remember that continuous learning and practice are essential to becoming proficient in PHP development.