PHP – Theory (Exam-Focused)

1) Basics & Syntax

- PHP runs on server; files end with `.php`.
- Code blocks: `<?php ... ?>`
- Echo/print: `echo "Hi";` / `print("Hi");`
- Comments: `//`, `#`, `/* ... */`
- Case-sensitive for variables; functions are not.

2) Variables, Types, Constants

- `$x = 10; $name = "A"; $flag = true;`
- Types: int, float, string, bool, array, object, resource, null.
- Dynamic typing; `var_dump($x)` shows type.
- Constant: `define('PI', 3.14);` or `const PI = 3.14;`

3) Operators & Control Flow

- Arithmetic, comparison (== vs ===), logical (`&&`, `||`, `!`), null coalescing `??`, spaceship `<=>`.
- `if/elseif/else`, `switch`, `match` (PHP 8), loops: `for`, `while`, `do...while`, `foreach`.

4) Strings & Arrays

- String funcs: `strlen`, `strtolower`, `strpos`, `substr`, `str_replace`, `trim`, `explode`, `implode`.
- Arrays: indexed, associative, multidimensional.
- Array funcs: `count`, `array_push`, `array_merge`, `array_map`, `array_filter`, `array_reduce`, `sort`, `ksort`, `usort`.

5) Functions & Scope

- `function add($a, $b = 0): int { return $a + $b; }`
- Pass by value/ref: `function f(&$x){}`
- Anonymous fn/closures: `$f = fn($x) => $x*$x;`
- Include/require: `include`, `require`, `*_once`.

6) Superglobals & Forms

- `$_GET`, `$_POST`, `$_REQUEST`, `$_SERVER`, `$_FILES`, `$_SESSION`, `$_COOKIE`, `$_ENV`.
- Validate & sanitize: `filter_input(INPUT_POST, 'email', FILTER_VALIDATE_EMAIL)`.

7) Sessions & Cookies

- Session: `session_start(); $_SESSION['user']='A';`
- Cookie: `setcookie('token','abc', time()+3600, '/');`

8) Files & Uploads

- Read/write: `file_get_contents`, `file_put_contents`, `fopen/fgets/fwrite`.
- Uploads: `$_FILES['file']['name/temp_name/size']`, move via `move_uploaded_file`.

9) OOP in PHP

- Class, object, visibility: `public/protected/private`.
- Inheritance, `final`, `static`, abstract classes & interfaces, traits (`use LogTrait;`).
- Namespaces & autoload (`spl_autoload_register`).
- Magic methods: `__construct`, `__destruct`, `__get`, `__set`, `__toString`.

10) PDO & MySQL (Prepared)

```
$pdo = new PDO('mysql:host=localhost;dbname=test','root','',[
  PDO::ATTR_ERRMODE=>PDO::ERRMODE_EXCEPTION
]);
$stmt = $pdo->prepare("SELECT * FROM users WHERE email = ?");
$stmt->execute([$email]);
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

11) Errors & Exceptions

- `try { ... } catch (Exception $e) { echo $e->getMessage(); } finally {}`
- Error levels, `error_reporting(E_ALL); ini_set('display_errors',1);`

12) Dates/Times

- `date('Y-m-d H:i:s')`, `DateTime`, `DateInterval`, `DateTimeZone`.

13) Security Essentials

- SQLi → use **prepared statements**.
- XSS → **escape output** (`htmlspecialchars`).
- CSRF → **tokens** in forms.
- Passwords → `password_hash`, `password_verify`.
- File upload validation (mime, size, extension).
- Disable dangerous functions in production; least-privilege DB user.

14) Composer & PSR

- `composer init`, `composer require monolog/monolog`.

- PSR-4 autoload in `composer.json`.

15) PHP 8+ Sweet Stuff

- Union types `function f(int|float $x) {}`,
- Attributes `#[ORM\Entity]`,
- Constructor property promotion,
- Nullsafe `?->`,
- `match` expression.

---

25 PHP PYQs with Answers (Part 1 of 100)

Q1. Difference between `echo` and `print`?

Ans: Both output strings. `echo` is slightly faster and can take multiple args; `print` returns `1` (usable in expressions) and takes one arg.

Q2. What is the output type of `var_dump()`?

Ans: It **prints** (doesn't return) detailed type & value info of variables.

Q3. Show how to define and use a constant.

```
<?php
define('APP_NAME','MyApp');
const VERSION = '1.0';
echo APP_NAME.' v'.VERSION;
```

Q4. Explain == vs ===.

Ans: == compares values with type juggling; === compares **value and type** (strict).

Q5. Write a function with default arg and return type.

```
function greet(string $name = 'Guest'): string {
  return "Hello, $name";
}
```

Q6. Create and iterate an associative array.

```
$marks = ['Ali'=>88,'Bea'=>92];
foreach ($marks as $name=>$m) echo "$name: $m\n";
```

Q7. Convert CSV string to array and back.

```
$s = "red,green,blue";
$arr = explode(',', $s);
$back = implode(';', $arr); // "red;green;blue"
```

Q8. Sort an associative array by keys ascending.

```
$prices = ['b'=>20,'a'=>10,'c'=>30];
ksort($prices);
```

Q9. Filter array to keep even numbers only.

```
$nums = [1,2,3,4,5,6];
$even = array_filter($nums, fn($x)=>$x%2===0);
```

Q10. What is a closure? Example.

```
$mult = 3;
$f = function($x) use ($mult) { return $x*$mult; };
echo $f(5); // 15
```

Q11. Sanitize and validate email from POST.

```
$email = filter_input(INPUT_POST,'email',FILTER_SANITIZE_EMAIL);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) { /* invalid */ }
```

Q12. Start a session and store username.

```
session_start();
$_SESSION['user'] = 'nora';
```

Q13. Set & read a cookie for 1 hour.

```
setcookie('theme','dark', time()+3600, '/'); // set
$theme = $_COOKIE['theme'] ?? 'light';       // read
```

Q14. Secure file upload (core steps).

```
if (isset($_FILES['pic']) && $_FILES['pic']['error']==UPLOAD_ERR_OK) {
  $tmp = $_FILES['pic']['tmp_name'];
  $name = basename($_FILES['pic']['name']);
```

```
  $ext = strtolower(pathinfo($name, PATHINFO_EXTENSION));
  $allowed = ['jpg','png','webp'];
  if (in_array($ext,$allowed) && mime_content_type($tmp) === 'image/png') {
    move_uploaded_file($tmp, __DIR__."/uploads/$name");
  }
}
```

Note: Check size, MIME, generate random filename, never trust client name.

Q15. Simple class with constructor & method.

```
class Box {
  public function __construct(private int $w, private int $h) {}
  public function area(): int { return $this->w * $this->h; }
}
echo (new Box(5,6))->area(); // 30
```

Q16. Interface and class implementation.

```
interface Logger { public function log(string $m): void; }
class EchoLogger implements Logger {
  public function log(string $m): void { echo "[LOG] $m"; }
}
```

Q17. Use a Trait to share behavior.

```
trait Timestamps {
  public function now(): string { return date('c'); }
}
class Post { use Timestamps; }
```

Q18. Connect to MySQL with PDO and insert safely.

```
$pdo = new PDO('mysql:host=localhost;dbname=app','user','pass');
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$stmt = $pdo->prepare("INSERT INTO users(name,email) VALUES(?,?)");
$stmt->execute([$name,$email]);
```

Q19. Fetch all rows as assoc array.

```
$stmt = $pdo->query("SELECT id,name FROM users");
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

Q20. Hash and verify passwords.

```
$hash = password_hash($plain, PASSWORD_DEFAULT);
if (password_verify($plain, $hash)) { /* ok */ }
```

Q21. Prevent XSS when outputting user input.

```
echo htmlspecialchars($userInput, ENT_QUOTES, 'UTF-8');
```

Q22. Generate a CSRF token and verify.

```
// Generate
session_start();
$_SESSION['csrf'] = bin2hex(random_bytes(32));
// Include in form as hidden input
// Verify
if (!hash_equals($_SESSION['csrf'], $_POST['csrf'] ?? '')) die('CSRF!');
```

Q23. Use `match` (PHP 8) instead of switch.

```
$type = 'pdf';
$mime = match($type){
  'png' => 'image/png',
  'jpg','jpeg' => 'image/jpeg',
  'pdf' => 'application/pdf',
  default => 'application/octet-stream'
};
```

Q24. Autoload classes with `spl_autoload_register`.

```
spl_autoload_register(function($class){
  $path = __DIR__.'/src/'.str_replace('\\','/',$class).'.php';
  if (file_exists($path)) require $path;
});
```

Q25. Build a tiny JSON API endpoint.

```
header('Content-Type: application/json');
$data = ['ok'=>true,'time'=>date('c')];
echo json_encode($data);
```

Q26. How do you read a text file line by line in PHP?

```
$fp = fopen("data.txt", "r");
while (($line = fgets($fp)) !== false) {
    echo $line."<br>";
}
fclose($fp);
```

Q27. Write PHP code to append text to a file.

```
file_put_contents("log.txt", "New line\n", FILE_APPEND);
```

Q28. Explain the difference between `include` and `require`.

Ans:

- `include` → gives a warning if file missing, script continues.
- `require` → fatal error if missing, script stops.

Q29. How to upload a file in PHP safely?

Steps:

1. Check `$_FILES` for errors.
2. Validate MIME type and size.
3. Generate a random filename.
4. Use `move_uploaded_file()`.

Q30. How do you check if a file exists?

```
if (file_exists("config.php")) { include "config.php"; }
```

Q31. What is output buffering in PHP?

Ans: It stores output in memory before sending to browser.

```
ob_start();
echo "Hello";
$content = ob_get_clean(); // capture output
```

Q32. Show how to create JSON from an array.

```
$arr = ["name"=>"Ali","age"=>21];
echo json_encode($arr);
```

Q33. How to decode JSON to array?

```
$json = '{"a":1,"b":2}';
$arr = json_decode($json, true);
```

Q34. How to redirect user to another page?

```
header("Location: login.php");
exit;
```

Q35. Difference between `GET` and `POST` methods?

- `GET`: data in URL, length limited, less secure.
- `POST`: data in request body, secure for forms, no size limit.

Q36. How to connect MySQLi (procedural)?

```
$conn = mysqli_connect("localhost","root","","test");
if (!$conn) die("Error: ".mysqli_connect_error());
```

Q37. Difference between MySQLi and PDO?

- **MySQLi** → works only with MySQL.
- **PDO** → supports many databases, prepared statements, OO interface.

Q38. Perform a SELECT query using MySQLi.

```
$res = mysqli_query($conn,"SELECT id,name FROM users");
while ($row = mysqli_fetch_assoc($res)) {
    echo $row['id']." ".$row['name']."<br>";
}
```

Q39. Perform UPDATE query with PDO prepared statement.

```
$stmt = $pdo->prepare("UPDATE users SET name=? WHERE id=?");
$stmt->execute(["Zara",5]);
```

Q40. How to handle exceptions in PHP?

```
try {
  throw new Exception("Error found!");
} catch (Exception $e) {
  echo $e->getMessage();
}
```

Q41. What is the difference between `isset()` and `empty()`?

- `isset($x)` → true if variable exists & not null.
- `empty($x)` → true if variable not set or false/0/""/null.

Q42. Explain `require_once` vs `include_once`.

- `_once` ensures file is included only once, avoiding redeclaration errors.

Q43. Demonstrate PHP session destroy.

```
session_start();
session_unset();
session_destroy();
```

Q44. How to encrypt data with `password_hash()`?

```
$hash = password_hash("mypwd", PASSWORD_BCRYPT);
```

Q45. Generate a random secure token.

```
$token = bin2hex(random_bytes(16));
```

Q46. Difference between `md5()` and `password_hash()`?

- `md5()` → fast, insecure for passwords.
- `password_hash()` → bcrypt/argon2, salted & secure.

Q47. How to handle file uploads larger than 2MB?

- Change `php.ini` → `upload_max_filesize`, `post_max_size`.
- Always validate size in script.

Q48. Explain difference between == and === with example.

```
var_dump(0 == "0");  // true
var_dump(0 === "0"); // false (different types)
```

Q49. Explain MVC architecture in PHP.

- **Model** → data & business logic.
- **View** → presentation (HTML).

- **Controller** → handles requests, connects model & view.
  Frameworks like Laravel, CodeIgniter follow MVC.

Q50. Show use of namespaces.

```
namespace App\Utils;
class Helper { public static function greet(){ echo "Hi"; } }
\App\Utils\Helper::greet();
```

Q51. What are prepared statements and why are they important?

Ans:
Prepared statements separate SQL from data, preventing SQL injection and improving performance for repeated queries.

```
$stmt = $pdo->prepare("SELECT * FROM users WHERE email=?");
$stmt->execute([$email]);
```

Q52. How to paginate database results?

```
$limit = 10;
$page = $_GET['page'] ?? 1;
$offset = ($page-1)*$limit;
$stmt = $pdo->prepare("SELECT * FROM posts LIMIT ? OFFSET ?");
$stmt->bindValue(1,$limit,PDO::PARAM_INT);
$stmt->bindValue(2,$offset,PDO::PARAM_INT);
$stmt->execute();
```

Q53. How do you send email in PHP?

```
mail("to@example.com","Subject","Message","From: me@example.com");
```

(In practice, use PHPMailer or SMTP for reliability.)

Q54. Explain difference between `require` and `autoload`.

- **require** → loads file explicitly.
- **autoload** → loads class automatically when used (via `spl_autoload_register`).

Q55. Create a simple login script (concept).

```
if ($_SERVER['REQUEST_METHOD']=='POST') {
  $stmt = $pdo->prepare("SELECT * FROM users WHERE email=?");
  $stmt->execute([$_POST['email']]);
  $user = $stmt->fetch();
  if ($user && password_verify($_POST['pass'],$user['password'])) {
    $_SESSION['uid']=$user['id'];
  } else { echo "Invalid"; }
}
```

Q56. What are JWTs (JSON Web Tokens)?

Ans:
JWT is a compact, signed token used for authentication.
Structure: `Header.Payload.Signature` (Base64).

Q57. How to handle file download in PHP?

```
header("Content-Type: application/pdf");
header("Content-Disposition: attachment; filename=report.pdf");
readfile("report.pdf");
```

Q58. What is Cross-Site Request Forgery (CSRF)?

Ans:
A malicious site tricks a logged-in user into performing unwanted actions.
Prevention: CSRF tokens, SameSite cookies.

Q59. How to prevent SQL Injection in PHP?

- Use prepared statements with `?` placeholders.
- Validate and sanitize input.

Q60. Explain difference between `array_merge()` and `+` operator.

- `array_merge()` → merges and reindexes keys.
- `+` → union of arrays; keys from left are preserved.

Q61. How to implement logout in PHP?

```
session_start();
session_unset();
```

```
session_destroy();
header("Location: login.php");
```

Q62. What is XSS (Cross-Site Scripting)?

Ans:
Injection of malicious scripts into webpages.
Prevention: `htmlspecialchars()`, CSP headers.

Q63. How do you upload multiple files?

```
foreach ($_FILES['photos']['tmp_name'] as $i=>$tmp) {
  move_uploaded_file($tmp, "uploads/".$_FILES['photos']['name'][$i]);
}
```

Q64. What is Composer in PHP?

Ans: Dependency manager for PHP.
Commands:

- `composer init`
- `composer require vendor/package`
- Autoload via `vendor/autoload.php`

Q65. Explain PSR standards.

Ans:

- **PSR-1/2/12** → coding style.
- **PSR-4** → autoloading standard.
- **PSR-7** → HTTP messages.

Q66. What is MVC? Name PHP frameworks using it.

Ans:
MVC = Model View Controller architecture.
Frameworks: Laravel, Symfony, CodeIgniter, Yii.

Q67. Explain PHP error levels.

- `E_NOTICE` → minor issue.
- `E_WARNING` → non-fatal.
- `E_ERROR` → fatal.
- Config: `error_reporting(E_ALL);`

Q68. What is difference between `unlink()` and `unset()`?

- `unlink("file.txt")` → deletes file.
- `unset($var)` → deletes variable.

Q69. How to use Traits in PHP?

```
trait Logger { function log($m){ echo $m; } }
class Test { use Logger; }
```

Q70. Difference between abstract class and interface?

- Abstract class → can have implemented + abstract methods.
- Interface → only method signatures (until PHP 8, now may have defaults).

Q71. How to perform cURL request in PHP?

```
$ch = curl_init("https://api.example.com");
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$res = curl_exec($ch);
curl_close($ch);
```

Q72. Explain difference between `public`, `protected`, `private`.

- `public` → accessible everywhere.
- `protected` → within class & subclasses.
- `private` → within same class only.

Q73. Demonstrate exception chaining.

```
try {
  try { throw new Exception("Inner"); }
  catch (Exception $e) { throw new Exception("Outer",0,$e); }
} catch (Exception $e) {
  echo $e->getPrevious()->getMessage(); // "Inner"
}
```

Q74. How to schedule jobs in PHP?

Ans: Use **Cron jobs** in Linux or Task Scheduler in Windows to run PHP scripts periodically.

Q75. How to handle JSON API request & response?

```
$data = json_decode(file_get_contents("php://input"), true);
header("Content-Type: application/json");
echo json_encode(["ok"=>true,"received"=>$data]);
```

Q76. What is caching in PHP and why is it used?

Ans:
Caching stores frequently used data in memory/disk to reduce DB calls and speed up response.
Examples: **OPcache**, Memcached, Redis.

Q77. How to enable OPcache in PHP?

Ans: In `php.ini`:

```
opcache.enable=1
opcache.memory_consumption=128
```

Q78. Explain difference between session and JWT authentication.

- **Session** → server stores session ID & data.
- **JWT** → self-contained token stored on client; scalable for APIs.

Q79. How to secure a REST API in PHP?

- Use HTTPS
- JWT or OAuth 2.0 for auth
- Rate limiting
- Input validation & sanitization

Q80. How to implement file-based caching in PHP?

```
$key = md5("page1");
$cache = "cache/$key.html";
if (file_exists($cache) && time()-filemtime($cache)<60) {
  readfile($cache);
  exit;
}
ob_start();
// dynamic content
$html = ob_get_clean();
file_put_contents($cache, $html);
```

Q81. What is the purpose of `.htaccess` in PHP projects?

Ans: Configure Apache settings: URL rewriting, redirects, access control, error pages.

Q82. How to rewrite URLs with `.htaccess`?

```
RewriteEngine On
RewriteRule ^product/([0-9]+)$ product.php?id=$1 [L]
```

Q83. Explain dependency injection in PHP.

Ans: Technique to provide required objects (dependencies) from outside rather than creating them inside class. Improves testability & maintainability.

Q84. Demonstrate using PHP with AJAX.

```
// JS
fetch("data.php").then(r=>r.text()).then(console.log);
// data.php
echo "Hello from PHP";
```

Q85. What are PSR-7 HTTP messages?

Ans: Standard interface for HTTP requests & responses (used in frameworks like Slim, Laravel).

Q86. How to use PHP sessions across subdomains?

Set cookie domain:

```
session_set_cookie_params(['domain'=>'.example.com']);
session_start();
```

Q87. Explain PHP's garbage collection.

Ans: PHP automatically frees memory of unused variables. Cyclic references are cleared by Garbage Collector (enabled via `gc_enable()`).

Q88. How to implement role-based access control (RBAC)?

- Store user role in DB/session.
- Check before action:

```
if ($_SESSION['role']!=='admin') die("Forbidden");
```

Q89. What is a design pattern? Name a few in PHP.

Ans: A reusable solution to common problems in OOP design.
Examples: Singleton, Factory, Strategy, Observer, MVC.

Q90. Show example of Singleton in PHP.

```
class DB {
  private static $inst;
  private function __construct() {}
  public static function get() {
    return self::$inst ??= new DB();
  }
}
```

Q91. How to upload images and generate thumbnails?

Use GD/Imagick:

```
$img = imagecreatefromjpeg("big.jpg");
$thumb = imagescale($img,150,150);
imagejpeg($thumb,"thumb.jpg");
```

Q92. How to send JSON response with HTTP status?

```
http_response_code(201);
header("Content-Type: application/json");
echo json_encode(["created"=>true]);
```

Q93. What is difference between REST and SOAP in PHP?

- **REST** → lightweight, JSON, stateless.
- **SOAP** → XML-based, strict, WS-* standards.

Q94. How to consume REST API in PHP?

```
$res = file_get_contents("https://api.github.com");
```

or with **cURL** for more control.

Q95. What is PHPUnit?

Ans: A unit testing framework for PHP. Used to test functions/classes automatically.

Q96. Show example PHPUnit test case.

```
class MathTest extends PHPUnit\Framework\TestCase {
  public function testAdd() {
    $this->assertEquals(4, 2+2);
  }
}
```

Q97. How to handle environment variables in PHP?

Use `.env` file with `vlucas/phpdotenv` package:

```
$dotenv = Dotenv\Dotenv::createImmutable(__DIR__);
$dotenv->load();
echo $_ENV['DB_USER'];
```

Q98. What is difference between `static` and `self` in PHP?

- `self::` → refers to current class only.
- `static::` → late static binding (refers to subclass if extended).

Q99. How to deploy PHP app securely?

- Disable `display_errors` in production.
- Use HTTPS.

- Apply least-privilege DB user.
- Harden `php.ini` (`disable_functions`).
- Use firewall, WAF, monitoring.

Q100. Summarize PHP 8+ key features.

- Union types
- Nullsafe operator `?->`
- Named arguments
- Attributes/annotations
- JIT compilation
- Match expression