PHP: 100 PYQS WITH ANSWERS (FULL)

Each slide = 1 Question + Answer/Code

Q1 – Q100

Q1. DIFFERENCE BETWEEN ECHO AND PRINT?

- echo: faster, multiple args.
  print: returns 1, single arg.

Q2. WHAT DOES VAR_DUMP() DO?

- Prints type + value details of variables.

Q3. HOW TO DEFINE/USE CONSTANTS?

- define('PI',3.14);
  const PI = 3.14;

Q4. == VS ===?

- == value only; === value + type (strict).

Q5. FUNCTION WITH DEFAULT ARG + RETURN TYPE?

- function add(int $a,int $b=0): int { return $a+$b; }

Q6. LIST PHP SCALAR + SPECIAL TYPES.

- int, float, string, bool, array, object, resource, null.

Q7. START SESSION & STORE DATA.

- session_start(); $_SESSION['user']='Ali';

Q8. COOKIES VS SESSIONS?

- Cookie: client-side; Session: server-side.

Q9. FOREACH EXAMPLE.

- foreach($arr as $k=>$v){ echo "$k:$v\n"; }

Q10. MYSQLI CONNECT (PROCEDURAL).

- $c=mysqli_connect('localhost','root','','test');

Q11. INDEXED VS ASSOCIATIVE ARRAYS?

- Indexed: numeric keys; Associative: string keys.

Q12. ARRAY_MERGE USAGE.

- $c=array_merge($a,$b);

Q13. COUNT ARRAY ELEMENTS.

- count($arr);

Q14. MULTIDIMENSIONAL ARRAY EXAMPLE.

- $a=[[1,2],[3,4]];

Q15. ITERATE WITH KEY/VALUE.

- foreach($a as $k=>$v){ /*...*/ }

Q16. COMMON STRING FUNCS.

- strlen, strtolower, strtoupper, strpos, substr.

Q17. SUBSTR EXAMPLE.

- substr('abcdef',1,3) // 'bcd'

Q18. EXPLODE VS IMPLODE.

- explode: string→array; implode: array→string.

Q19. TRIM USAGE.

- trim(' hi ') // 'hi'

Q20. STRING INTERPOLATION.

- $name='Ali'; echo "Hello $name";

Q21. $_GET VS $_POST.

- GET in URL; POST in body; POST safer for forms.

Q22. $_REQUEST MEANING.

- Union of GET+POST+COOKIE (avoid in strict apps).

Q23. BASIC FILE UPLOAD.

- move_uploaded_file($_FILES['f']['tmp_name'],'uploads/a.txt');

Q24. FILE READ/WRITE QUICK.

- file_put_contents('a.txt','hi'); echo file_get_contents('a.txt');

Q25. INCLUDE VS REQUIRE.

- include: warning on miss; require: fatal on miss.

Q26. READ FILE LINE BY LINE.

- while(($l=fgets($fp))!==false){ echo $l; }

Q27. APPEND TO FILE.

- file_put_contents('log.txt','New\n',FILE_APPEND);

Q28. OUTPUT BUFFERING.

- ob_start(); echo 'Hi'; $s=ob_get_clean();

Q29. SAFE UPLOAD STEPS.

- Check error, size, MIME, ext; use move_uploaded_file.

Q30. FILE_EXISTS USAGE.

- if(file_exists('config.php')) require 'config.php';

Q31. JSON ENCODE.

- echo json_encode(['a'=>1]);

Q32. JSON DECODE ASSOC.

- $arr=json_decode($json,true);

Q33. REDIRECT HEADER.

- header('Location: login.php'); exit;

Q34. GET VS POST ADVANTAGES.

- GET: bookmarkable; POST: larger+safer for sensitive data.

Q35. MYSQLI SELECT + FETCH.

- $r=mysqli_query($c,'SELECT * FROM t');
  while($row=mysqli_fetch_assoc($r)){}

Q36. MYSQLI VS PDO?

- MySQLi: MySQL-only; PDO: multi-DB, nicer prepared statements.

Q37. PDO CONNECT + ERRMODE.

- $pdo=new PDO('mysql:host=localhost;dbname=x','u','p',[PDO::ATTR_ERRMODE=>PDO::ERRMODE_EXCEPTION]);

Q38. PDO PREPARED SELECT.

- $st=$pdo->prepare('SELECT * FROM u WHERE email=?');
  $st->execute([$e]);

Q39. PDO UPDATE PREPARED.

- $st=$pdo->prepare('UPDATE u SET n=? WHERE id=?');
  $st->execute([$n,$id]);

Q40. TRY/CATCH EXAMPLE.

- try{ /*...*/ }catch(Exception $e){ echo $e->getMessage(); }

Q41. ISSET VS EMPTY.

- isset: set & not null; empty: false for '',0,null,[], not set.

Q42. REQUIRE_ONCE VS INCLUDE_ONCE.

- Both avoid duplicate load; require_once fatal on failure.

Q43. DESTROY SESSION (LOGOUT).

- session_start(); session_unset(); session_destroy();

Q44. PASSWORD_HASH USAGE.

- $h=password_hash($pwd,PASSWORD_DEFAULT);

Q45. PASSWORD_VERIFY USAGE.

- if(password_verify($pwd,$hash)){ /* ok */ }

Q46. MD5 VS PASSWORD_HASH.

- md5 fast/insecure; password_hash salted bcrypt/argon2.

Q47. LARGE UPLOAD LIMITS.

- php.ini: upload_max_filesize, post_max_size; check size in code.

Q48. == VS === EXAMPLE.

- var_dump(0=='0'); // true
  var_dump(0==='0'); // false

Q49. MVC MEANING.

- Model: data; View: UI; Controller: request handling.

Q50. NAMESPACES EXAMPLE.

- namespace App; class A{}; \App\A;

Q51. PREPARED STATEMENTS—WHY?

- Prevent SQLi; faster for repeated execs.

Q52. PAGINATION WITH LIMIT/OFFSET.

- $limit=10;$off=($p-1)*$limit; SELECT ... LIMIT ? OFFSET ?

Q53. SEND EMAIL (BASIC).

- mail('to@x.com','Subj','Msg','From: me@x.com');

Q54. AUTOLOAD VS REQUIRE.

- Autoload loads classes on demand; require is manual include.

Q55. SIMPLE LOGIN FLOW.

- Fetch by email; verify password; set $_SESSION['uid'].

Q56. JWT BASICS.

- Header.Payload.Signature; stateless auth token.

Q57. FORCE FILE DOWNLOAD.

- header('Content-Type: app/pdf'); header('Content-Disposition: attachment; filename=a.pdf'); readfile('a.pdf');

Q58. CSRF & PREVENTION.

- Use CSRF tokens; SameSite cookies; verify on POST.

Q59. PREVENT SQL INJECTION.

- Always use prepared statements + input validation.

Q60. ARRAY_MERGE VS + OPERATOR.

- merge reindexes; + keeps left keys, ignores dup keys.

Q61. PROPER LOGOUT REDIRECT.

- session_destroy(); header('Location: login.php');

Q62. XSS & PREVENTION.

- Escape output: htmlspecialchars($s,ENT_QUOTES,'UTF-8'); CSP.

Q63. MULTIPLE FILE UPLOAD.

- loop over $_FILES['f']['tmp_name'] and move each.

Q64. COMPOSER BASICS.

- composer init; composer require vendor/pkg; require 'vendor/autoload.php';

Q65. PSR STANDARDS QUICK.

- PSR-1/12 coding; PSR-4 autoload; PSR-7 HTTP messages.

Q66. MVC FRAMEWORKS NAMES.

- Laravel, Symfony, CodeIgniter, Yii, Slim (micro).

Q67. PHP ERROR LEVELS.

- E_NOTICE, E_WARNING, E_ERROR; error_reporting(E_ALL);

Q68. UNLINK VS UNSET.

- unlink deletes file; unset removes variable/array key.

Q69. TRAIT USAGE.

- trait Log{function log($m){echo $m;}} class A{ use Log; }

Q70. ABSTRACT VS INTERFACE.

- Abstract: some impl; Interface: method signatures (PHP8: defaults ok).

Q71. BASIC CURL GET.

- $ch=curl_init('https://api.example.com'); curl_setopt($ch,CURLOPT_RETURNTRANSFER,true); $r=curl_exec($ch); curl_close($ch);

Q72. PUBLIC/PROTECTED/PRIVATE.

- public everywhere; protected class+children; private class only.

Q73. EXCEPTION CHAINING.

- throw new Exception('Outer',0,new Exception('Inner'));

Q74. CRON JOB IDEA.

- Use OS cron to run php script periodically (e.g., */5 * * * * php job.php).

Q75. JSON API HANDLER.

- $data=json_decode(file_get_contents('php://input'),true); echo json_encode(['ok'=>true,'data'=>$data]);

Q76. WHAT IS CACHING?

- Store frequent data to speed up (OPcache, Redis, Memcached).

Q77. ENABLE OPCACHE.

- php.ini: opcache.enable=1; opcache.memory_consumption=128

Q78. SESSION VS JWT AUTH.

- Session: server state; JWT: self-contained token (scales).

Q79. SECURE REST API TIPS.

- HTTPS, JWT/OAuth2, rate limit, validate input, sanitize output.

Q80. FILE-BASED CACHING SNIPPET.

- if(fresh($f)) readfile($f); else { ob_start(); /*render*/ $h=ob_get_clean(); file_put_contents($f,$h);}

Q81. .HTACCESS PURPOSE.

- Apache per-dir config: rewrites, redirects, deny/allow, errors.

Q82. REWRITE PRETTY URL.

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

Q83. DEPENDENCY INJECTION IDEA.

- Give dependencies from outside; improves testability.

Q84. AJAX WITH PHP.

- JS fetch('data.php').then(r=>r.text()); PHP: echo 'Hi';

Q85. PSR-7 MEANING.

- Standard interfaces for HTTP req/resp objects.

Q86. SESSIONS ACROSS SUBDOMAINS.

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

Q87. GARBAGE COLLECTION.

- Cyclic GC clears ref cycles; gc_enable(), gc_collect_cycles().

Q88. RBAC QUICK CHECK.

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

Q89. COMMON DESIGN PATTERNS.

- Singleton, Factory, Strategy, Observer, MVC.

Q90. SINGLETON SAMPLE.

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

Q91. CREATE IMAGE THUMBNAIL (GD).

- $img=imagecreatefromjpeg('big.jpg');
  $t=imagescale($img,150,150);
  imagejpeg($t,'thumb.jpg');

Q92. JSON WITH STATUS CODE.

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

Q93. REST VS SOAP.

- REST: JSON, stateless; SOAP: XML, strict, WS-*.

Q94. CONSUME REST IN PHP.

- $res=file_get_contents('https://api.github.com'); // or cURL

Q95. WHAT IS PHPUNIT?

- Unit testing framework for PHP.

Q96. PHPUNIT TEST EXAMPLE.

- class T extends PHPUnit\Framework\TestCase{function testX(){ $this->assertEquals(4,2+2); }}

Q97. ENV VARS WITH DOTENV.

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

Q98. STATIC VS SELF.

- self:: binds to current class; static:: late static binding (subclass).

Q99. SECURE DEPLOYMENT TIPS.

- Disable display_errors, HTTPS, least-priv DB user, WAF, monitor.

Q100. PHP 8+ KEY FEATURES.

- Union types, nullsafe ?->, named args, attributes, JIT, match.