SEMESTRE IV · PRV001 · 90 HORAS

Programación Avanzada

Clase 1 — Introducción a PHP 8.5

Unidad I · Estructuras de datos y algoritmos aplicados

Técnico de Nivel Superior en Informática con especialidad en Programación

CFT Estatal de O'Higgins

CUATRO HORAS, NUEVE TRAMOS

La ruta de hoy

18:30
Piedra Rosetta
Python y PHP lado a lado

19:00 TALLER
Taller 1
Traducir cuatro ejercicios en 3v4l

19:30
Las cinco traiciones
Donde PHP no se porta como Python

19:50
El array de PHP
Lista y diccionario en una estructura

20:25
Break
Quince minutos

20:40
Funciones tipadas
La firma como contrato

21:00 TALLER
Taller 2
Un caso por persona — RA1

21:45
POO en PHP
Encapsular la pila del taller

22:05 TALLER
Puesta en común
Exposición y preguntas técnicas

Todo el código de hoy corre en 3v4l.org — sin instalar nada.

CONTEXTO

¿Por qué PHP en una asignatura avanzada?

1

Es el lenguaje del curso

PHP 8.5, publicado el 20 de noviembre de 2025. Trabajaremos con Laravel Herd, PhpStorm y GitHub.

2

Tiene tipado que se hace cumplir

Con strict_types activado, los tipos no son documentación: son un contrato que el motor verifica al ejecutar.

3

Cubre las cuatro unidades

Estructuras de datos, POO, concurrencia y pruebas: todo el programa se puede recorrer en PHP.

Ustedes ya saben programar. Lo de hoy no es aprender a programar de nuevo: es mudarse de lenguaje.

TRAMO 1 · 18:30

Piedra Rosetta: lo mismo, dos veces

Python 3 — lo que ya saben

```python
nombre = "Ana"
notas  = [6.0, 5.5, 4.8]

for n in notas:
    print(f"Nota: {n}")

def promedio(ns: list) -> float:
    return sum(ns) / len(ns)

print(nombre, promedio(notas))
```

PHP 8.5 — lo mismo, otra sintaxis

```php
<?php
declare(strict_types=1);

$nombre = "Ana";
$notas  = [6.0, 5.5, 4.8];

foreach ($notas as $n) {
    echo "Nota: {$n}\n";
}

function promedio(array $ns): float {
    return array_sum($ns) / count($ns);
}
```

Lo único realmente nuevo: `$` en cada variable · `;` al final de cada instrucción · llaves en vez de indentación · el archivo abre con `<?php`

TRAMO 3 · 19:30

Las cinco traiciones

Aquí es donde la intuición de Python los va a hacer perder tiempo.

EN PYTHON	EN PHP	LA TRAMPA
`"a" + "b"`	**`"a" . "b"`**	El + en PHP suma o une arrays: nunca concatena texto.
`==` compara con tipo	**`==` hace juggling**	"0" == false es verdadero. Usar siempre === y !==.
`elif` / `and` / `not`	**`elseif` / `&&` / `!`**	and y or existen, pero con otra precedencia. No usarlos.
`d["x"]` lanza KeyError	**`$d["x"]` avisa y da null**	El programa NO se detiene: sigue corriendo con null adentro.
la función lee globales	**la función no ve nada**	Todo lo que la función necesite entra por parámetro.

La cuarta es la que más los va a morder: en Python el programa se cae, en PHP sigue con un null adentro.

ACTIVIDAD · 19:00 — 30 MINUTOS

Taller 1 · Traducción dirigida

Traduzcan a PHP estos cuatro programas que ya saben escribir en Python:

1. Promedio de una lista de notas
2. Contar cuántas veces aparece cada palabra en un texto
3. Filtrar de una lista los elementos que cumplen una condición
4. Recorrer un diccionario imprimiendo clave y valor

Los datos van escritos dentro del código: en 3v4l no hay input()

Cómo trabajamos

HERRAMIENTA

3v4l.org, sin cuenta ni instalación

FORMATO

Individual, con consulta libre entre ustedes

EVIDENCIA

Peguen sus enlaces de 3v4l en el chat del curso

CUIDADO

El código en 3v4l queda público. Nada de datos reales

TRAMO 1 · EL DETALLE QUE IMPORTA

strict_types: el tipo deja de ser un adorno

En Python los type hints son documentación: nadie los verifica. En PHP, con strict_types activado, el motor los hace cumplir al ejecutar.

```php
<?php
declare(strict_types=1);   // primera línea, siempre

function promedio(array $ns): float {
    return array_sum($ns) / count($ns);
}

promedio("6.0");           // TypeError
promedio([6.0, 5.5]);      // 5.75
```

¿Por qué insistir?

- El error aparece en la línea que lo causó, no diez pasos después.
- La firma de la función se vuelve un contrato legible.
- Es lo que hace que una asignatura llamada avanzada tenga sentido en PHP.

Sin esa primera línea, PHP convierte "6.0" a número por su cuenta y el programa sigue — hasta que un día no sigue.

TRAMO 4 · 19:50 · ENTRAMOS A LA UNIDAD I

Un solo array hace de lista y de diccionario

En Python son dos tipos distintos. En PHP son el mismo: cómodo y peligroso a la vez.

Como lista

```php
$notas = [6.0, 5.5, 4.8];

$notas[] = 7.0;
echo count($notas);
// claves 0, 1, 2, 3
```

Como diccionario

```php
$stock = [
  "arroz"  => 12,
  "azúcar" => 3,
];
echo $stock["arroz"];
```

Recorriendo ambos

```php
foreach ($stock as $k => $v) {
  echo "{$k}: {$v}\n";
}

// clave => valor
```

El precio de esa comodidad

Nada distingue a simple vista una lista de un diccionario: hay que leer el código que lo llena. Y las claves numéricas no siempre quedan ordenadas ni consecutivas después de borrar elementos.

TRAMO 4 · CONTENIDO DEL PROGRAMA

Pila y cola, con el array que ya tienen

PILA · LIFO

```php
$pila = [];

$pila[] = "pagina1";      // apilar
$pila[] = "pagina2";

$ultima = array_pop($pila);
// devuelve "pagina2"
```

Sale el último que entró. Ejemplo: historial de navegación, deshacer.

COLA · FIFO

```php
$cola = [];

$cola[] = "cliente1";     // encolar
$cola[] = "cliente2";

$primero = array_shift($cola);
// devuelve "cliente1"
```

Sale el primero que entró. Ejemplo: atención de clientes, impresión.

!

Guárdenlo para la próxima clase

array_pop saca del final y no toca nada más. array_shift saca del inicio y reindexa todo el array. Hacen lo mismo pero no cuestan lo mismo, y esa diferencia es el tema de eficiencia básica del RA1.

ACTIVIDAD · 21:00 — 45 MINUTOS

Taller 2 · Un caso por persona

Cada uno toma un caso, elige la estructura y después la defiende ante el curso.

1

Historial de navegación

El botón Atrás debe devolver siempre la última página visitada.

2

Atención de clientes

Se atiende en estricto orden de llegada, sin adelantarse nadie.

3

Registro de usuarios

Buscar los datos de alguien a partir de su RUT, sin recorrer todo.

Qué tiene que entregar cada uno

1

Analizar qué operaciones pide el problema

2

Elegir la estructura y fundamentarla

3

Implementarla en PHP con datos de prueba

4

Enlace de 3v4l funcionando

TRAMO 6 · 20:40

La firma de la función es un contrato

```php
function buscarStock(
    array $inventario,
    string $producto
): ?int {
    return $inventario[$producto] ?? null;
}

$stock = ["arroz" => 12];

buscarStock($stock, "arroz");   // 12
buscarStock($stock, "azúcar");  // null, sin warning
```

`array $inventario`

El tipo va antes del nombre, al revés de Python.

`: ?int`

Devuelve un entero o null. El ? lo hace explícito.

`?? null`

Reemplaza la clave faltante por un valor, sin warning.

Leyendo solo la primera línea ya se sabe qué entra, qué sale y qué puede fallar. Eso es código mantenible.

TRAMO 8 · 21:45 · ASOMO A LA UNIDAD II

La misma pila, ahora encapsulada

```
class Pila {
    private array $items = [];

    public function apilar(string $item): void {
        $this->items[] = $item;
    }

    public function desapilar(): ?string {
        return array_pop($this->items);
    }
}

$p = new Pila();
$p->apilar("pagina1");
```

Contra lo que ya vieron en Python

`self` → `$this`

Y no se declara como parámetro del método.

`obj.x` → `$obj->x`

La flecha reemplaza al punto.

`private` de verdad

En Python el guion bajo es un acuerdo. Aquí el motor lo impide.

Nadie fuera de la clase puede tocar $items. La pila deja de ser un array suelto y pasa a ser una estructura que se defiende sola.

BONUS · EL LENGUAJE SIGUE VIVO

Tres cosas que PHP 8.5 trajo

Versión publicada el 20 de noviembre de 2025. No las usaremos hoy: son para que vean que esto no es un lenguaje congelado.

`array_first()` y `array_last()`

Devuelven el primer o último valor de un array, o null si está vacío.

```
$ultimo = array_last($eventos);

// antes hacía falta preguntar
// si el array estaba vacío
```

Operador pipe `|>`

Encadena funciones de izquierda a derecha, en vez de anidarlas hacia adentro.

```
$slug = $titulo
  |> trim(...)
  |> strtolower(...);
```

`clone` con cambios

Copiar un objeto cambiando propiedades, sin reconstruirlo entero.

```
return clone($this, [
  'alpha' => $alpha,
]);
```

Fuente: php.net/releases/8.5 · 3v4l.org ya ejecuta PHP 8.5, así que se pueden probar hoy mismo.

PUESTA EN COMÚN · 22:05

Defiendan su decisión técnica

Cada uno expone su caso del Taller 2 y responde estas tres preguntas.

1. ¿Qué características del problema fueron clave para seleccionar la estructura de datos?

2. ¿Qué dificultades surgirían si se utilizara una estructura inadecuada?

3. ¿Cómo influye la forma de acceso a los datos en la eficiencia de la solución?

Las tres preguntas vienen textuales del programa de la asignatura, Unidad I.

CIERRE

Lo de hoy dentro del semestre

RA1	**Estructuras de datos y algoritmos**	Hoy: pila, cola y diccionario sobre arrays.
RA2	**Programación orientada a objetos**	Hoy: el asomo con la clase Pila.
RA3	**Concurrencia y asincronía**	Más adelante en el semestre.
RA4	**Pruebas, depuración y control de versiones**	Por eso la tarea de hoy es abrir GitHub.

Cuatro horas y ya tocamos dos de los cuatro resultados de aprendizaje del semestre.

PARA LA PRÓXIMA CLASE

Tres cosas

1

Crear cuenta en GitHub

La vamos a necesitar sí o sí en la Unidad IV. Háganlo hoy mismo.

2

Intentar instalar en casa

Laravel Herd y PhpStorm. Si algo falla, traíganme el mensaje de error.

3

Dejar sus enlaces de 3v4l

Los cuatro del Taller 1 y el del caso que les tocó.

Próxima clase: por qué array_shift cuesta más que array_pop, y cómo medirlo.