

# Laboratorio II / Laboratorio de Software II 2024

---

## Clase Teórica 02: Condicional Simple, Doble y Múltiple, Funciones predefinidas



**Docente: Myriam Ruiz**  
Licenciada en Informática  
Profesora en Computación y Matemática  
Programadora Universitaria



# Condicional

---

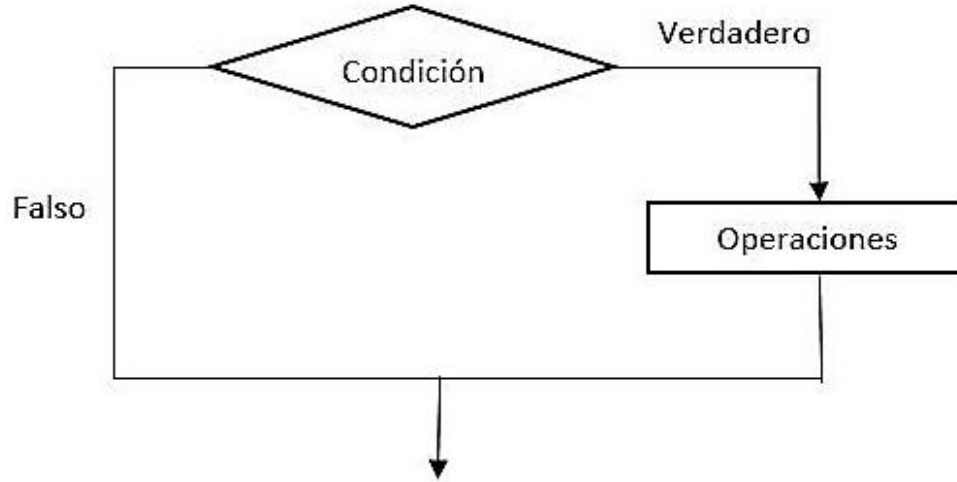
- En la vida diaria tenemos que tomar decisiones, al momento de cruzar la calle, al decidir qué comer o qué ponernos.
- En Programación esta toma de decisiones se implementa con las **Estructuras de Selección**.



# Condicional Simple

---

Cuando sólo nos interesa saber si una condición es cierta, usaremos el **Condicional Simple**



# Condicional Simple

---

## Algoritmo

```
SI (Condición) ENTONCES  
    A1  
Fin-Si
```

## PHP

```
if (Condición) {  
    A1;  
}
```

# Condicional Simple - Ejemplo

---

## Enunciado

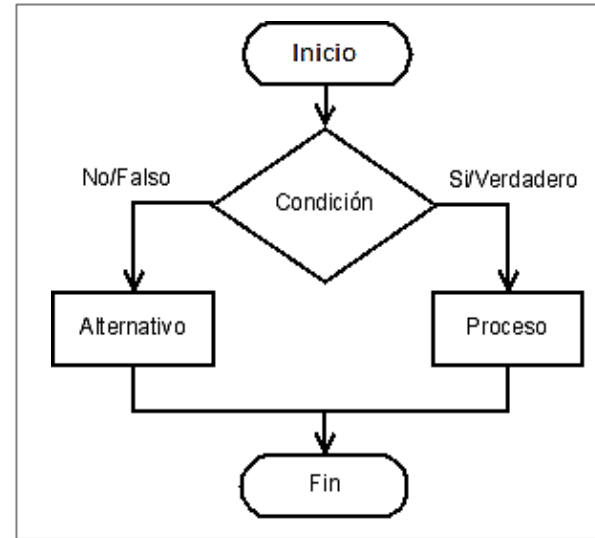
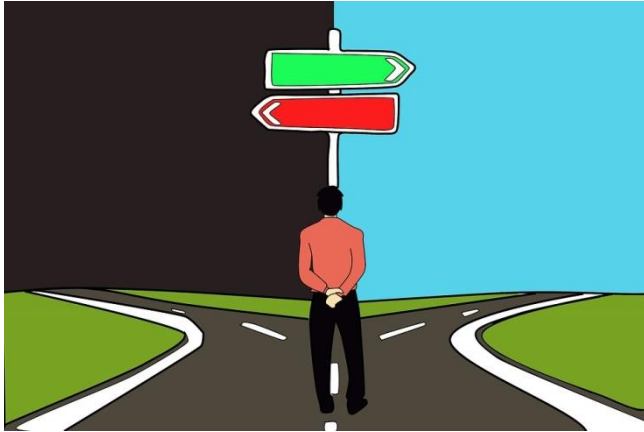
Si un ciudadano tiene 16 años o más, entonces  
Escribir el mensaje puede votar

## PHP

```
$edad = 20;  
if ($edad >= 16) {  
    echo '<p>Puede votar</p>';  
}
```

# Condicional Doble

Cuando tenemos dos posibles respuestas ante una condición (pregunta), podremos usar un **Condicional Doble**



# Condicional Doble

---


## Algoritmo

```
SI (Condición) ENTONCES
    Acciones cuando se
    cumple la condición
SINO
    Acciones cuando NO se
    cumple
FIN-SI
```

## PHP

Ingresar si en condición  
el if recibe un valor  
verdadero (true)

```
if (Condición) {  
    Acciones cuando se  
    cumple la condición;  
} else {  
    Acciones cuando NO se  
    cumple;  
}
```



# Condicional Doble – Cuál rama se mostrará?

---

## Ejemplo 1

```
<article>
  <?php
    $valor = true;
    if ($valor) {
      echo '<p>Ingresó un true</p>';
    } else {
      echo '<p>Ingresó un false</p>';
    }
  }
```

## Ejemplo 2


```
$valor = false;
if ($valor) {
  echo '<p>Ingresó un true</p>';
} else {
  echo '<p>Ingresó un false</p>';
}
```



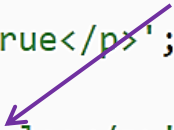
# Condicional Doble – Cuál rama se mostrará?

---

```
<article>
  <?php
    $valor = true;
    if ($valor) {
      echo '<p>Ingresó un true</p>';
    } else {
      echo '<p>Ingresó un false</p>';
    }
  }
```



```
$valor = false;
if ($valor) {
  echo '<p>Ingresó un true</p>';
} else {
  echo '<p>Ingresó un false</p>';
}
```



# Operadores de Comparación I

---

- Los condicionales utilizan **operadores de Comparación**, que permiten comparar el contenido de una variable contra otro valor.
- Leerlos siempre de izquierda a derecha.
- Si se cumplen devuelven 1 (true) y sino 0 (false):
- $a > b$  (\$a es Mayor que \$b)
- $a < b$  (\$a es menor que \$b)
- $a \geq b$  (\$a es Mayor o igual que \$b)
- $a \leq b$  (\$a es menor o igual que \$b)


# Operadores de Comparación II

---

- `$a == $b` (\$a es igual a \$b, sin importar el tipo de dato)
- `$a === $b` (\$a es idéntico a \$b, deben tener mismo tipo)
- `$a != $b` (\$a es distinto, no es igual a \$b)
- `$a !== $b` (\$a no es idéntico a \$b, puede ser porque los valores no coincidan y también porque no tengan el mismo tipo de dato)

# Operadores de Comparación - Ejemplos

Son iguales, dio un resultado 1



Es una forma en que se muestra true

```
<article>
  <?php
    $num1 = 4;
    $num2 = '4';
    $valor = ($num1 == $num2); // se pregunta por valor no por tipo
    if ($valor) { // podría ser if ($num1 == $num2)
      echo '<p>Son iguales, dio un resultado ' . $valor . '</p>';
    } else {
      echo '<p>Son distintos, dio un resultado ' . $valor . '</p>';
    }
  ?>
</article>
```

# Operadores de Comparación - Ejemplos

Son distintos, dio un resultado       



El falso no se muestra

```
<article>
  <?php
    $num1 = 4;
    $num2 = '4';
    $valor = ($num1 === $num2); // se pregunta si coincide valor y tipo
    if ($valor){ // podría ser if ($num1 === $num2)
      echo '<p>Son iguales, dio un resultado ' . $valor . '</p>';
    } else {
      echo '<p>Son distintos, dio un resultado ' . $valor . '</p>';
    }
  ?>
</article>
```

# Condicional Doble - Ejemplo

---

## Enunciado


Dado el estado del tiempo,  
Si el tiempo está cálido  
Escribir vamos al río, sino  
Escribir Vamos al  
shopping.

## Resultado para

`$tiempo = 'lluvioso'` y `$tiempo`  
`= 'cálido' ?`

## PHP

```
$tiempo = 'frío';  
if ($tiempo == 'cálido') {  
    echo '<p>Vamos al río</p>';  
} else {  
    echo '<p>Vamos al shopping</p>';  
}
```



# Condicional Doble Sintetizado con un Operador Ternario

---

**Operador Ternario:** Puede sintetizar un condicional doble

\$resultado = (condición)?'resp. afirmativa':'resp. negativa';

# Condicional Doble Sintetizado con un Operador Ternario

## Condicional Doble

```
$entrada = 200;  
$dinero = 1500;  
if ($dinero > $entrada) {  
    $salida = 'Salgo! Wiii';  
} else {  
    $salida = 'No salgo';  
}  
echo '<p>' . $salida . '</p>';
```

## Operador Ternario

```
$entrada = 200;  
$dinero = 1500;  
$salida = ($dinero > $entrada)? 'Salgo! Wiii': 'No salgo';  
echo '<p>' . $salida . '</p>';
```





# Condicional if elseif

---

## PHP

elseif permite tener código más sintético, pero trabaja igual que else if

```
if (Condición1) {
```

Acciones cuando se cumple la condición1;

```
} elseif (condición2) {
```

Acciones cuando se cumple la condición2;

```
} else {
```

Acciones cuando NO se cumple condición 1 ni 2;

```
}
```

# Condicional elseif - Ejemplo

---

## Enunciado de Ejemplo

Dadas las notas de parcial de un alumno, calcule su promedio y muestre un mensaje de acuerdo a las siguientes reglas: Si el promedio es mayor o igual a 7 entonces promociona, sino, si es mayor o igual a 4 entonces regulariza, sino recupera.

# Condicional if elseif else

**PHP de Ejemplo:** Dadas las notas de un alumno si su promedio es mayor o igual a 7 mostrar el mensaje Promociona, Sino si es mayor a 4 mostrar Regular, sino mostrar Recupera

```
<article>
    <?php
        $priParc = mt_rand(0,10);
        $segParc = mt_rand(0,10);
        $promedio = ($priParc + $segParc)/2;
        if ($promedio >= 7) {
            echo '<p>Nota = ' . $promedio . ' -> Promociona</p>';
        } elseif ($promedio >= 4) {
            echo '<p>Nota = ' . $promedio . ' -> Regular</p>';
        } else {
            echo '<p>Nota = ' . $promedio . ' -> Recupera </p>';
        }
    ?>
</article>
```

# Operadores Lógicos

---

- Usaremos los Operadores Lógicos para poder establecer más de una condición.
- `! $a`            No, Negado: devuelve true si \$a no es true.
- `$a && $b`        Y: devuelve true si tanto \$a como \$b son true.
- `$a || $b`        O: devuelve true si \$a o \$b (o ambos) es true.
- Hay más...

<https://www.php.net/manual/es/language.operators.logical.php>

# Operadores Lógicos – Ejemplo I

```
<article>
  <?php
    $p = true;
    $q = true;
    $w = $p && !$q; // $p AND negado $q
    $x = $p && $q;
    $y = !$p || !$q; // negado $p OR negado $q
    $z = !$p || $q;
    echo '<p>p = ' . $p . '</p>';
    echo '<p>q = ' . $q . '</p>';
    echo '<p>w = ' . $w . '(los false no se muestran) </p>';
    echo '<p>x = ' . $x . '</p>';
    echo '<p>y = ' . $y . '(los false no se muestran) </p>';
    echo '<p>z = ' . $z . '</p>';
```

# Operadores Lógicos – Ejemplo I

---

## **Ejemplo operadores &&, || y !**

p = 1

q = 1

w = (los false no se muestran)

x = 1

y = (los false no se muestran)

z = 1

# Operadores Lógicos – Ejemplo I

---

```
if ($w) { // controla si es verdadero
|   echo '<p>w = ' . $w . '</p>';
|
| }
else {
|   echo '<p>w = false </p>'; // para mostrar cuando es false
| }
}
```

w = false

# Operadores Lógicos – Evaluación short-circuit

---

- La evaluación de Circuito corto, evaluación mínima, evaluación perezosa o evaluación McCarthy, consiste en que:
  - En el **&&** la segunda condición no se evalúa si la primera condición evaluada da falso.
  - En el **||** si la primera condición se evalúa y el resultado es verdadero, ya no se evalúa la segunda condición. Sólo cuando la primera condición es falsa se evalúa la segunda.



# Algunas funciones predefinidas

---

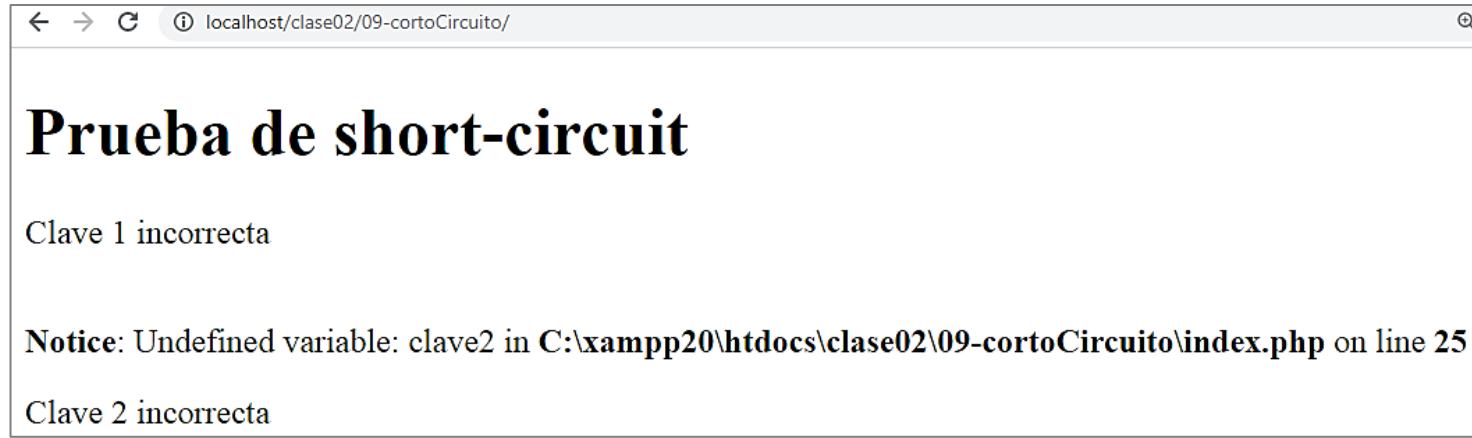
- Para el manejo de variables
  - `isset ( mixed $var [, mixed $... ] ) : bool`
    - Pregunta si una variable (o varias), existe y no es nula, devuelve true o false

# Operadores Lógicos – Evaluación short-circuit

```
section>
  <article>
    <?php
      $usuario = 'admin'; // no se define $clave1
      if ((isset($clave1)) && ($clave1 == 'admin123')){
        echo '<p>Clave 1 correcta</p>';
      } else {
        echo '<p>Clave 1 incorrecta</p>';
      }
    ?>
    <?php
      $usuario = 'admin'; // no se define $clave2
      if (($clave2 == 'admin123') && (isset($clave2))){
        echo '<p>Clave 2 correcta</p>';
      } else {
        echo '<p>Clave 2 incorrecta</p>';
      }
    ?>
  </article>
```

Funcionan igual los  
códigos?

# Operadores Lógicos – Evaluación short-circuit



Entonces, importará el orden de las preguntas cuando es un Evaluación short-circuit?

Si! Ya que poner primero la condición con el **isset** nos protege de producir un error al querer usarla si no existe

## Evaluación de Circuito corto – Ejemplo con ||

```
<article>
  <?php
    $laboratorio = 'Aprobada';
    $elementos = 'No aprobada';
    $carrera = 'Licenciatura en Informática';
    if (($elementos == 'Aprobada'
        || $carrera == 'Licenciatura en Informática')
        && $laboratorio == 'Aprobada'){
      echo '<p>Es de Licenciatura en Informática
        o no aprobó Elementos, pero Aprobó Laboratorio -> Puede Cursar</p>';
    } else {
      echo '<p>No puede cursar Laboratorio II</p>';
    }
  ?>
</article>
```

Es de Licenciatura en Informática o no aprobó Elementos, pero Aprobó Laboratorio -> Puede Cursar

# Jerarquía de Operadores

---

!

\*, / , %

+, -, .

<, <= , > ,>=

==, !=, ===, !== , <>

&&

||

? :

=, +=, -=, \*= ,

\*\*=, /=, .-=, %-= , &=

# Operadores Lógicos – Ejemplo II

← → ↻ ⓘ localhost/clase02/06-logicos2/

## Ejemplo operadores && y ===

Acceso denegado

Por qué?

```
<article>
  <?php
    $usuario = 'admin';
    $clave = '1234';
    // la condición se cumple cuando ambas condiciones sea verdad
    if($usuario === 'admin' && $clave === 1234){
      echo '<p>Acceso otorgado</p>';
    }
    else{
      echo '<p>Acceso denegado</p>';
    }
  ?>
</article>
```

# Estructura SEGUN

---

- Estructura SEGUN se utiliza...



- Cuando ante una pregunta existen más de dos respuestas posibles, que no se pueden manejar de manera eficiente con una estructura Si.

# switch (SEGUN)

---

- PHP implementa el SEGÚN con la instrucción **switch**

```
SEGUN variable
    valor1:
        respuesta1
    valor2:
        respuesta2
...
SINO
    respuesta_por_defecto
Fin-SEGUN
```

```
switch ($variable o expresión) {
    case valor1:
        respuesta1;
        break;
    case valor2:
        respuesta2;
        break;
    ...
    default:
        respuesta_por_defecto;
}
```



# switch – Ejemplo

---

- Según el color de camiseta ingresado, el script adivinará el equipo al que pertenece



# switch – Ejemplo

<article>

```
<?php
```

```
    $color = 'azul';
```

```
    $respuesta = '';
```

```
    switch ($color) {
```

```
        case 'rojo':
```

```
            $respuesta = 'River, San Martín, Newells o San Lorenzo';
```

```
            break;
```

```
        case 'celeste':
```

```
            $respuesta = 'Atlético o Racing';
```

```
            break;
```

```
        case 'azul':
```

```
            $respuesta = 'Boca o San Lorenzo';
```

```
            break;
```

```
        default:
```

```
            $respuesta = 'No conozco esa camiseta';
```

```
    }
```

```
    echo '<p>Color: ' . $color . ' pertenece a ' . $respuesta . '</p>';
```

```
?>
```



localhost/clase02/07-equipos/

Color: azul pertenece a Boca o San Lorenzo

## switch – Ejemplo

---

- Importante: switch es sensible a mayúsculas y minúsculas, por lo que si escribimos case 'Azul' o case 'AZUL', no encontrará coincidencia.

# switch – Si más de una opción tiene misma respuesta

---

- Cuando más de una opción recibe la misma respuesta, se puede escribir así:

```
<?php
    switch ($variable o expresión){
        case valor1:
        case valor2:
        case valor3:
            respuesta1;
            break;
        case valor4:
            respuesta2;
            break;
        ...
        default:
            respuesta_por_defecto;
    }
?>
```

## switch – Ejemplo con números

---

- En un súper, si compra un producto paga el precio normal
- Pero si compra 2, 3 o 4, paga con 10% de descuento cada uno
- Con 5 o 6, paga 20% de descuento cada uno

## switch – Ejemplo 02

- En un super se otorga un descuento del 10% si se compran entre 2 y 4 productos iguales o del 20% si se compra 5 o 6, decidir cuál es el costo final por producto

```
<?php
    include 'html/header.html';
    $cant = 2; // cantidad de productos (para probar el código)
    $costoUnitario = 100;
    $costoFinal = 0;
    switch($cant){
        case 1:
            $costoFinal = $costoUnitario;
            break;
        case 2:
        case 3:
        case 4:
            $costoFinal = $costoUnitario*0.9; // 10% menos
            break;
```

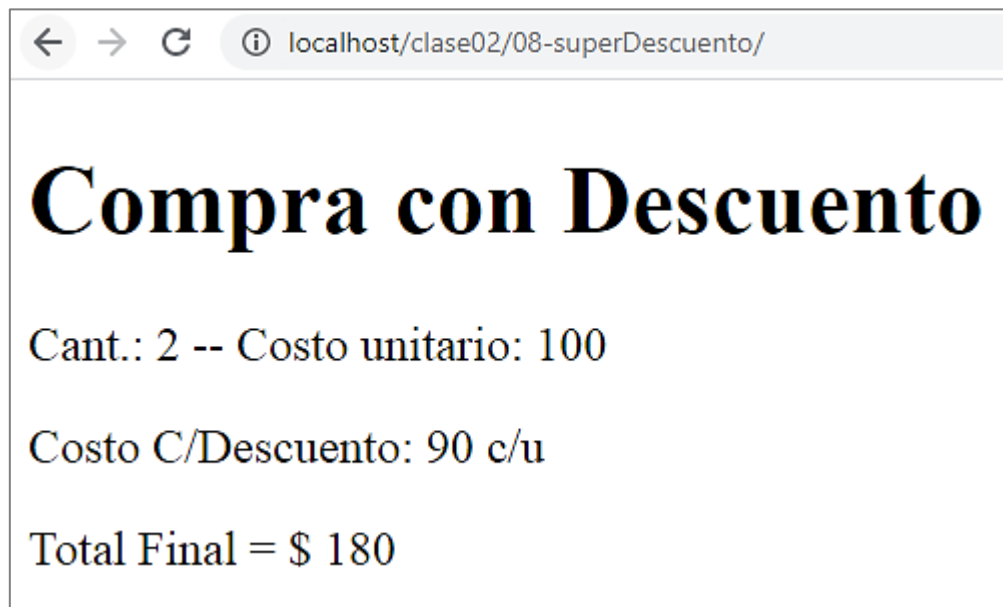
## switch – Ejemplo 02

---

```
case 5:
case 6:
    $costoFinal = $costoUnitario*0.8; // 20% menos
}
$total = $costoFinal * $cant;
echo '<p>Cant.: ' . $cant . ' -- Costo unitario: ' . $costoUnitario . '</p>'
echo '<p>Costo C/Descuento: ' . $costoFinal . ' c/u</p>';
echo '<p>Total Final = $ ' . $total . '</p>';
include 'html/footer.html';
?>
```

## switch – Ejemplo 02

---





# Funciones - Concepto

---

- ¿Qué son las funciones?
  - Son un grupo de instrucciones que tienen un objetivo en común
  - Se les asocia un nombre para poder invocarla / llamarla, las veces que haga falta
  - Pueden recibir (parámetros o argumentos) y retornar resultados

# Funciones – Para qué sirven

---

- Supongamos nuestro código necesita hacer varias veces ciertos cálculos, como obtener el Neto de un sueldo...

# Funciones – Para qué sirven

Observamos señalado por llaves el código que se repite

```
const JUBILACION = 0.11;
```

```
const LEY19032 = 0.03;
```

```
const OS = 0.03;
```

```
$sueldoBruto = 9800; ←
```

```
$desJubilacion = $sueldoBruto * JUBILACION;
```

```
$desLey = $sueldoBruto * LEY19032;
```

```
$desOS = $sueldoBruto * OS;
```

```
$sueldoNeto = $sueldoBruto - ($desJubilacion + $desLey + $desOS);
```

```
echo 'Sueldo Neto: $' . number_format($sueldoNeto,0,',','.' );
```

```
// otros códigos... ←
```

```
echo '<hr>';
```

```
$sueldoBruto = 7600; ←
```

```
$desJubilacion = $sueldoBruto * JUBILACION;
```

```
$desLey = $sueldoBruto * LEY19032;
```

```
$desOS = $sueldoBruto * OS;
```


```
$sueldoNeto = $sueldoBruto - ($desJubilacion + $desLey + $desOS);
```

```
echo 'Sueldo Neto: $' . number_format($sueldoNeto,0,',','.' );
```

# Funciones – Para qué sirven

Si en vez de repetir tantas veces el mismo código, usamos una función, el código queda más claro y prolijo, dejando el desarrollo de la función en otra parte del código.

```
$sueldoBruto = 9800;  
$sueldoNeto = calcularSueldoNeto($sueldoBruto);  
echo 'Sueldo Neto: $' . number_format($sueldoNeto,0,',','');  
// otros códigos  
echo '<hr>';  
$sueldoBruto = 7600;  
$sueldoNeto = calcularSueldoNeto($sueldoBruto);  
echo 'Sueldo Neto: $' . number_format($sueldoNeto,0,',','');
```



# Funciones – Para qué sirven

---

- ¿Por qué usar funciones?
  - Ahorrarnos la escritura de código, podemos reutilizar el mismo código varias veces.
  - Dividir tareas grandes en subtareas menos complejas ("Divide and Conquer", Divide y Vencerás).
  - Facilitar la comprensión y lectura del código.

# Funciones - Tipos

---

- Existen 2 tipos de Funciones:
  - **Predefinidas del lenguaje** (ya existen, sólo las usamos)
  - **Definidas por el usuario**
    - ¿Cuándo se necesitará crearlas?
      - ❖ Cuando queramos hacer algo que no se puede realizar con funciones predefinidas del lenguaje (se verá en otra clase)

# Algunas funciones predefinidas

---

- En la clase 01 se vio como mostrar un número con cierta cantidad de decimales con `number_format`, sin embargo lo que finalmente devuelve es texto.
- Hay funciones que recortan los decimales, dejando como resultado un número, algo que es útil si luego queremos operar con las variables

# Algunas funciones predefinidas

---

- Para Redondear

- `round ( float $val [, int $precision = 0 [, int $mode = PHP_ROUND_HALF_UP ] ] ) : float`
  - Recibe float y devuelve float con menos decimales indicados por \$precisión
- `ceil ( float $value ) : float`
  - Recibe float y devuelve float redondeado hacia arriba (sin cifras decimales)
- `floor ( float $value ) : mixed`
  - Recibe float y devuelve float redondeado hacia abajo (sin cifras decimales)




# Algunas funciones predefinidas

- Para Redondear

```
<article>
    <?php
        $radio = 10;
        $area1 = pi() * $radio**2;
        echo '<p>Area sin redondear = ' . $area1 . '</p>';
        echo '<p>Tipo de dato: ' . gettype($area1) . '</p><hr>';
        $area2 = round($area1, 3);
        echo '<p>Area redondeada con round = ' . $area2 . '</p>';
        echo '<p>Tipo de dato: ' . gettype($area2) . '</p><hr>';
        $area3 = ceil($area1); // redondea hacia el entero hacia arriba
        echo '<p>Area redondeada con ceil = ' . $area3 . '</p>';
        echo '<p>Tipo de dato: ' . gettype($area3) . '</p><hr>';
        $area4 = floor($area1); // redondea hacia el entero hacia abajo
        echo '<p>Area redondeada con floor = ' . $area4 . '</p>';
        echo '<p>Tipo de dato: ' . gettype($area4) . '</p>';
    ?>
```

# Algunas funciones predefinidas

- Para Redondear

	
<h2>Rendondeos</h2>	
Area sin redondear = 314.15926535898	
Tipo de dato: double	
Area redondeada con round = 314.159	
Tipo de dato: double	
Area redondeada con ceil = 315	
Tipo de dato: double	
Area redondeada con floor = 314	
Tipo de dato: double	

# Funciones Predefinidas: Que trabajan con ASCII

---

- `chr ( int $ascii ) : string`
  - Recibe un código ASCII y devuelve el carácter correspondiente
- `ord ( string $string ) : int`
  - Recibe un string (texto) y convierte la primera letra en un entero sin signo entre 0 y 255. Se recomienda usar con una letra y no con una frase

# Ejemplo: Letras aleatorias

- Una forma de crear letras aleatorias:
  - ✓ Trabajar con los Valores ASCII, dando un intervalo numérico para ciertas letras y usando la función `mt_rand($inferior, $superior)`
  - ✓ Luego se debe convertir el valor obtenido a carácter, para recién obtener una letra, utilizando la función `chr($número)`

Caracteres ASCII imprimibles		
32	espacio	64 @
33	!	65 A
34	"	66 B
35	#	67 C
36	\$	68 D
37	%	69 E
38	&	70 F
39	'	71 G
40	(	72 H
41	)	73 I
42	*	74 J
43	+	75 K
44	,	76 L
45	-	77 M
46	.	78 N
47	/	79 O
48	0	80 P
49	1	81 Q
50	2	82 R
51	3	83 S
52	4	84 T
53	5	85 U
54	6	86 V
55	7	87 W
56	8	88 X
57	9	89 Y
58	:	90 Z
59	;	91 [
60	<	92 \
61	=	93 ]
62	>	94 ^
63	?	95 _
		96 `
		97 a
		98 b
		99 c
		100 d
		101 e
		102 f
		103 g
		104 h
		105 i
		106 j
		107 k
		108 l
		109 m
		110 n
		111 o
		112 p
		113 q
		114 r
		115 s
		116 t
		117 u
		118 v
		119 w
		120 x
		121 y
		122 z
		123 {
		124
		125 }
		126 ~

# switch - Ejemplo 1 con letras

---

- Generar letras minúsculas e indicar cuáles son vocales y cuáles no

```
<article>
    <?php
        $miAscii = mt_rand(97, 122); // de 'a' hasta 'z'
        $letra = chr($miAscii); // convierte a letra un ascii
        switch($letra){
            case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u':
                echo '<p>' . $letra . ' es vocal</p>';
                break;
            default:
                echo '<p>' . $letra . ' no es vocal';
        }
    }
```

## switch - Ejemplo 2 con letras

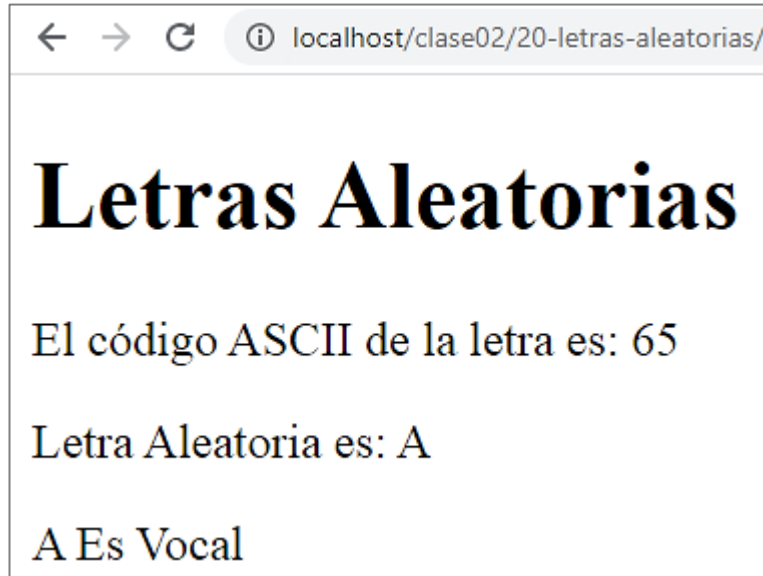
- Generar letras mayúsculas, pero tomando como límite las letras A y Z.

```
<article>
  <?php
    $codigoLetra = mt_rand(ord('A'), ord('Z'));
    echo '<p>El código ASCII de la letra es: ' . $codigoLetra . '</p>';
    echo '<p>Letra Aleatoria es: ' . chr($codigoLetra) . '</p>';
    $letra = chr($codigoLetra);
    switch ($letra) {
      case 'A':
      case 'E':
      case 'I':
      case 'O':
      case 'U':
        echo '<p> ' . $letra . ' Es Vocal</p>';
        break;
      default:
        echo '<p> ' . $letra . ' Es Consonante</p>';
    }
  ?>
```

# switch - Ejemplo con letras

---

- Generar letras mayúsculas, pero tomando como límite las letras A y Z.



# Artificial Intelligence (Inteligencia Artificial)

---

- La inteligencia artificial, o IA, es tecnología que permite que las computadoras simulen la inteligencia humana y las capacidades humanas de resolución de problemas.
- Por sí sola o combinada con otras tecnologías (por ejemplo, sensores, geolocalización, robótica), la IA puede realizar tareas que de otro modo requerirían inteligencia o intervención humana.



# Desarrollo guiado por AI

---

- Mirando hacia el futuro (muy cercano), el desarrollo de software guiado por AI promete transformar el rol de los programadores.
- Desde la generación de código a partir de descripciones en lenguaje natural hasta la optimización automática de algoritmos, la AI está abriendo nuevas fronteras en la creatividad y eficiencia para los devs.

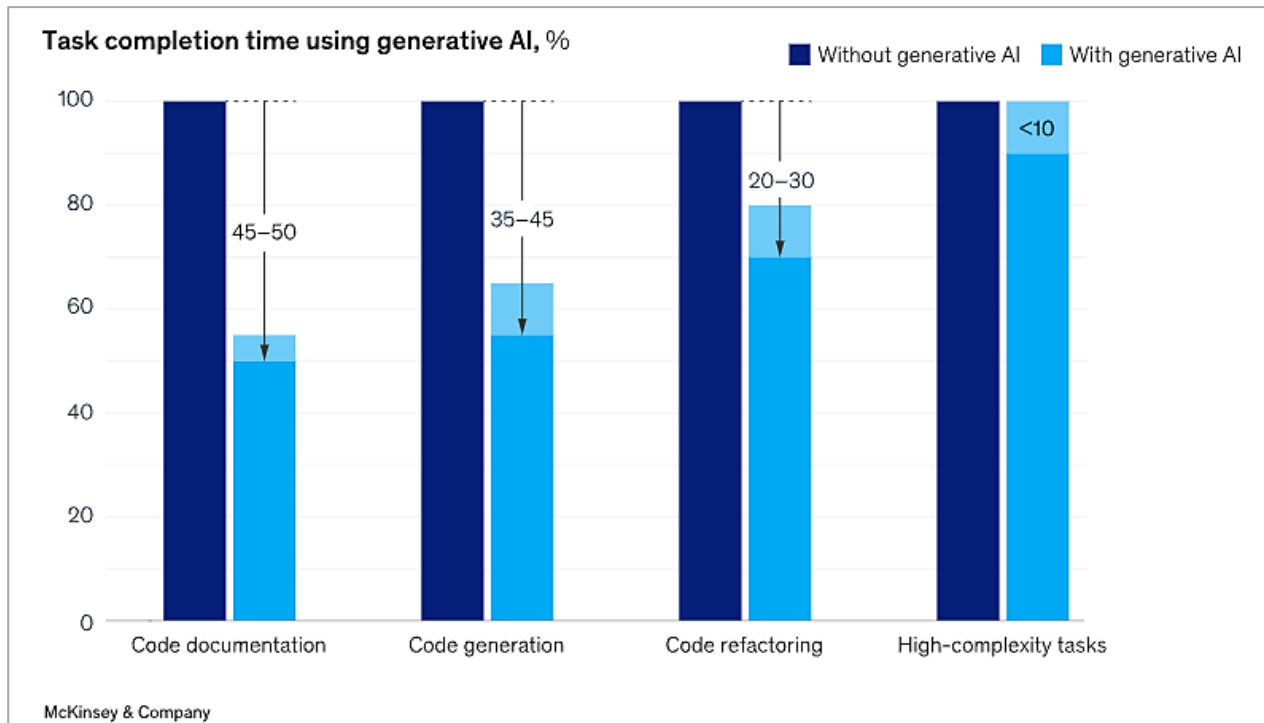
# Desarrollo guiado por AI

---

- La AI no pretende reemplazar a los programadores.
- Sino:
  - Potenciar sus habilidades.
  - Liberándolos de tareas más repetitivas.
  - Permitiéndoles centrarse en otros aspectos, como la innovación y el diseño de soluciones más complejas y creativas.

# Qué aporta la AI







La IA generativa puede incrementar la velocidad de desarrollo, pero menos para tareas complejas



# Qué aporta la AI

---

## Actividades genéricas en el desarrollo de software y uso

- Herramientas de diseño y modelado 
- Gestión de Requerimientos 
- Asistentes de programación 
- Validadores de Seguridad 
- Pruebas Automatizadas 
- Documentación 

# ¿Cómo crear un prompt efectivo?

---

- Debemos indicar:
  - **Contexto:** Brindar la información necesaria para que la IA entienda el escenario. Esto puede incluir detalles del tema, el objetivo de la respuesta o la situación específica, si debe actuar como un profesional de alguna área, etc.
  - **Instrucciones claras:** Indica explícitamente lo que deseas que la IA haga. Cuanto más claro y específico sea, mejor será la respuesta.
  - **Formato de respuesta deseado:** Indica cómo deseas recibir la respuesta, por ejemplo, si la respuesta debe ser en un formato de lista, en párrafos cortos, en un lenguaje técnico o coloquial, etc.
  - **Ejemplos (opcional):** Si es necesario, incluye ejemplos de respuestas o de lo que estás buscando para guiar a la IA.
  - **Limitaciones o restricciones (opcional):** Especifica cualquier cosa que debería evitarse, como temas o estilos particulares, tipo de instrucciones de un lenguaje de programación, etc.
  - **Tono o estilo (opcional):** Si es importante, especifica el tono o estilo de la respuesta, como formal, informal, amigable, profesional, etc.

# Malos hábitos (desventajas) de usar IA

---

- **Copiar y pegar sin razonar:** Muchos programadores principiantes tienden a copiar y pegar el código generado sin comprenderlo, lo que puede resultar en código de baja calidad y difícil de mantener.
- **No modularizar el código:** Al generar grandes cantidades de código de una sola vez, es común que los desarrolladores no lo dividan en módulos o archivos más pequeños, lo que dificulta su mantenimiento y comprensión.
- **Dependencia excesiva:** Algunos desarrolladores dependen completamente de las IA para resolver problemas, sin consultar la documentación oficial u otras fuentes que podrían proporcionar soluciones más actualizadas o precisas. Además, va en contra de la innovación a tomar soluciones que ya existen.
- **Evitar estudiar las bases:** Se corre el riesgo de que los programadores dejen de aprender y entender las bases fundamentales de la programación, confiando demasiado en las respuestas generadas por IA.
- **Incapacidad para reconocer código de baja calidad:** La falta de comprensión profunda del código puede llevar a los desarrolladores a aceptar soluciones no tan óptimas o inseguras sin darse cuenta de sus problemas.

# Hábito de Copiar y pegar sin razonar

---

- **Falta de comprensión profunda:** Cuando los desarrolladores copian y pegan código sin intentar entender cómo funciona, pierden la oportunidad de aprender los conceptos subyacentes. Esto significa que pueden completar tareas a corto plazo, pero no desarrollan las habilidades necesarias para resolver problemas más complejos en el futuro.
- **Código de baja calidad:** El código generado automáticamente puede no siempre seguir las mejores prácticas de programación. Si un programador simplemente copia y pega el código sin revisarlo o modificarlo, puede acabar con un código que es difícil de mantener, poco eficiente o incluso con errores ocultos que podrían causar problemas más adelante.
- **Dificultad para depurar y mantener:** Si un programador no comprende el código que ha copiado, será mucho más difícil para él depurar errores cuando algo sale mal. Además, cuando se necesiten cambios o mejoras en el futuro, la falta de comprensión del código existente puede hacer que estas tareas sean mucho más complicadas y propensas a errores.
- **Desarrollo de un "falso sentido de competencia":** Al completar proyectos utilizando código generado por IA, algunos programadores pueden llegar a creer que están mejorando sus habilidades, cuando en realidad solo están siguiendo instrucciones sin realmente entender lo que están haciendo. Esto puede dar lugar a una falsa confianza en sus habilidades, lo que podría ser perjudicial en entornos laborales más exigentes.