21-09-29/30 - Manipulación de datos

string

El problema de php es que tiene un sorporte UNICODE muy pobre, por motivos historicos.
Una cadena es una secuecnia de bytes, no una secuencia de caracteres.
El problema reside en UTF-8, donde un caracter puede ocupar más de un byte

¿Como escribir literales de tipo cadena?

4 formas:

- comillas dobles
- comillas simples
- heredoc
- nowdoc

1. comillas dobles: Suponen que lo que hay dentro que el interprete de php lo interpreta. Se lleva a cabo una interpretacion, por lo que habra una serie de caracteres que se sustituiran Un ejemplo son las secuencias de escape, que comienzan con backslash `\`. `\n` por ejemplo sólo funcionará en comillas dobles.

Interpolación (expansion de variables)

Otra cosa es la interpolación (expansión de variables). Consiste en meter una variable dentro de una cadena, haciendo que esta se sustituya por su valor

La regla de estilo dice que, siempre que no haga falta lo conrtario, las cadenas deben ir siempre en comillas simples. Usar solo comillas dobles cuando se necesite interpretar/interpolación.

es posible capar la secuencia de escapa, poniendo otro backslash. Tambien funcoina con variable, usando backslash delante del identificador

Podemos indicar una variable encerrándola entre llaves:

```
>>> $h = "pepe"
=> "pepe"
>>> echo "hola {$h}ola"
hola pepeola⏎
```

Indexación
en PHP no existe el tipo caracter, por lo que al usar indexacion ($cadena[0]), el indice se mueve por

bytes. Dado que algunos caracteres ocupan mas de un byte, a la hora de querer acceder a la indexación de un caracter especial, (la letra 'ó' de la palabra 'cami´on') no

con la indexacion tambien podemos alterar el caracter de una cadena, indicando su posición y asignandole una cadena que contenga un solo caracter (solo se espera una cadeana con una longitud de un byte).

Esta operación practicamente no la vamos a utilizar.

Debido al problema de que las cadenas solo se miden por secuencias de bytes, tenemos el modulo mbstring que nos proporciona una serie de funciones compatibles con caracteres multibyte.

strlen(string $string): la longitud de la cadena
trim: elimina espacios de una cadena
rtrim: elimina espacios por la deerecha
ltrim,: elimina espacios por la izquierda

- strpos: devuelve la posicion de la primera ocurrencia de un caracter en una cadena, false si no esncuentra ninguna. tiene un pparametro adiciona, offset, indicando la posicion desde la cual quieres comenzar a buscar. Por defecto su valor es 0. No se pueden usar valores negativos en el offset

mb-substr(cadena, posicion inicio, longitud)

Si yo quiero de "camión" la letra ó:
mb_substr("camión", 4, 1);

2. comilla simples

heredoc

heredoc permite escribir una cadena que ocupe varias lineas
Todo lo que haya dentro, a partir de los caracteres `<<<`, indicando con un marcador donde quieres que termine el texto. Aqui se respetaran los saltos de linea, y si dentro pones una variable se interpola, si tuviesemos secuancias de escape, se interpretará

```
>>> $nombre ='Pepe';
>>> $bloque = <<<EOT
Yo
me
llamo
$nombre
\n
EOT;
```

nowdoc

nowdoc es a las comillas simples, lo que heredoc es a las comillas dobles. No se realizaraña ningun tipo de interpretación o interpolación.
Para usarlo, el marcador que usabamos en nowdoc

4 tipos compuestos

- array
- object
- callable
- iterable

Finalmente dis tipos especiales:

- resource
- null

null

Tipo de dato, cuyo unico valor es null

si uso var_dump, me volcara a la salida el valor y el tipo de una expresión

is_null($x);

tambien podemos usar `$x == null`
pero esto nos provocara efectos indeseados, por ejemplo

```
>>> $x = null;
=> null
>>> is_null($x);
=> true
>>> $x == null;
=> true
>>> false == null;
=> true
>>> "" == null;
=> true
```

null siempre es convertido a string vacio.

Arrays

El tipo de datos mñas importante con diferencia. Mucho mas flexives que los arrays de java. Son muy parecidos a los dicionarios de python. Una especie de fusión entre ambos

> Los arrays en PHP son llamados *arrays asociativos*

Es una estrucutra de datos que contiene elementos, a los que se puede acceder mediante una clave. Cadad elemento de un array es una pareja clave-valor. Por lo que se trata de una colección de datos. Las claves pueden ser numeros enteros, o cadenas, pero siempre serán **únicas**

Con los numeros, lo que tendriamos es un array al estilo java, con la ventaja de que los arrays en php son dinamicos: puede crecer y encojerse en tiempo de ejecucion; no necesitamos determinar su tamaño de anteano. Ademas los tipos de los valores no hay que determinarlos de antemano, por lo que cada pareja clave-valor podra contener valores de distintos tipos.

Crear array

- forma antigua: La función array. recibe los elementos como parametros, sdeparados por comas.

`array('a', 'b', 'c')`

> Llamarlo posición sería incorrecto. La palabra adecuada para referirnos a estos *indice* es **clave**

- Forma moderna: Poner entre corchetes los elementos tal como queremos que vayan ordenados:

```
>>> $array =  ['a', 'b', 'c'];
=> [
     "a",
     "b",
     "c",
   ]
>>> $array[1];
=> "b"
>>> $array[1] = 'z';
=> "z"
>>> $array;
=> [
     "a",
     "z",
     "c",
   ]
>>>
```

Tambien podemos indicar maualmente cada clave y asignarla con `=>` al valor que deseemos:
`[0=>'a', 1=>'b', 2=>'c']`
También:
`[0=>'a', 'b', 'c']`

Todo elemento nuevo agregado se ubicará al final del array, y a php no le importa el orden de los elementos, ni si sus claves son correlativas.

Por eso podemos hacer esto:
`$a[] = 'm'`
de este modo, le decimos a php que agregue un nuevo elemtno, revisando el ultimo indice mas alto que se utilizo en el array, para usarlo sumandole 1. Usa una especie de variable que recuerda el entero más alto utilizado en dicho array.

tambien podemos usar:
`$a[-1] = 'y'`
Recordemos que esto no son indices como tal, son simplemente **claves**.

en definitiva, son diccionarios con la capacidad de compatibilizar con el concepto primitivo de arrays en C

eliminar elementos de un array

Usaremos la orden ya mencionada `unset`:
`unset($var[indice]);`

Comprobar si un elemento existe en un array

- con la construcción `isset`
  `isset($array[clave])`

El gran problema de isset es que devuelve true si la variable existe, y su valor es distinto de nulo. Para los casos en los que tenemos valores null en nuestro array podemos usar la funcion array_key_exists

- Con la función array_key_exists
  independientemente del valor que contenga la clave que solicitamos, esta solo se centrará en comprobar si existe la clave:
  `array_key_exists(clave, array);`

Buscar valores en un array

array_search devuelve la primera clave correspondiente en caso de exito, o false si no existe.

`array_search(valor, array)`

Buscar elementos a partir de su clave.

Esto lo haremos a traves de la indexacion

array_unique: elimina valores duplicados de un array

Entre todas las funciones disponibles para arrays en este lenguaje, encontramos algunas que devuelven una copia del array recibido, sienod asi funciones puras, y otras probocaran el estado lateral de modificar el array existente.

asort ordena un array, por valores devolviendo true si ha modificando el array original con exito, false en caso contrario.

ksort ordena un array, por claves devolviendo true si ha modificando el array original con exito, false en caso contrario.

array_combine realizará una combinación entre dos arrays:
`array_combine(arrays_keys($array1), arrays_values($array2))`

contar elementos de un array

`count($array)`

podemos comprobar si un array está vacío si hacemos:

empty($array) para comprobar si un array está vacío
empty funciona con otros tipos de datos, porque por motivos de conversión explicita podra devolver resultados inesperados:

`empty("0")` devolverá true, ya que por conversión de tipos, la cadena "0" se convertirá a boolean, y false se considera que no ocupa espacio, dando como resultado true.

30/09/2021 17:23

Dimensionalidad de los arrays

Esta claro que podemos contener a un array dentro de otro.

```
>>> $a = [4, [1, 2], 3];
>>> echo $a[1][1];
    => 2
```

Mostrar arrays

Podemos mostrar por var_dump, o por print_r

var_dump

```
>>> var_dump($a)
array(3) {
  [0]=>
  int(4)
  [1]=>
  array(2) {
    [0]=>
```

```
    int(1)
    [1]=>
    int(2)
  }
  [2]=>
  int(3)
}
=> null
```

print_r

```
>>> print_r($a)
Array
(
    [0] => 4
    [1] => Array
        (
            [0] => 1
            [1] => 2
        )

    [2] => 3
)
=> true
```

Constantes

Suelen definirse al comienzo del programa, siendo onligatorio para declararlas con la función `cons`.
No llevan el símbolo dolar como prefijo
Las constantes no se expanden (no son resueltas en el procesamiento de cadenas).
no pueden ser modificadas o eliminadas.
Se pueden declarar con la función `define($var, $value)`
define tiene dos parámetros:

- Nombre de la variable a declarar
- Valor que asignar

```
define('PI', 3.14);
=> true
```

Antes se usaba la funcion `cons`, que se podía usar solo con tipos de datos escalares (bool, int, float y string), pero con la llegada de `define` encontramos la posibilidad de realizar constantes de otros elementos, como por ejemplo arrays.

Constantes predefinidas

Ya comentamos que para el tipo de datos integer existián una serie de constantes que nos porporcionaba en tiempo real detalles sobre su rango de operatividad.

Del mismo modo encontraremos multitud de constantes que se inicializan al comienzo de arrancar un interprete de php. Estas podemos encontrarlas comenzando por `PHP_`

Comprobar si existen contantes

Con la función `defined($var)`.

```
>>> define('PI', 3.14);
=> true
>>> PI
=> 3.14
>>> defined('PI')
=> true
>>> defined('FOO')
=> false
```

Manipulación de datos

Comprobaciones

comprobación de tipos

- gettype()

- is_*()

Comprobación de valores

- is_numeric()

ctype_*()

Son funciones que devuelven cadenas y comprueba distintas características, como por ejemplo si una cadena solo contiene digitos (digit), o si solo contiene caracteres alfabéticos(alpha) ó alfanuméricos (alnum).

Converseiones

Tienen su mandanga. SAbiendo que es de tipado debil, muchas veces el lenguaje tratará de hacer conversiones ímplicitas mediante *casting*
Por supuesto podemos hacer conversines explicitas, con `(*) dato`
- implicitas
- explicitas

Conversion a numeros enteros

Conversion a cadena

booval
intval
floatval
strval

Conversion a null

> se parece al coalesce de Java (`$a ?? $b`)

Casi siempre una conversaion automática

- desde booleanos: flase 0, true 1
- desde numeros reales: trunca el valor, perdiendo la información decimal.

Comparaciones

< ; <= ; > ; >= ; !=

Equivalencias e identidad (== ; === ; !==)

- == : Operador de igualdad flexible.
- === : Operador de igualdad estricta. Compara tanto los datos como su tipo de dato.
- !== : Operador de igualdad estricta invertida.

Otros tipos de comparaciones (<=> ; ??)

- <=> : Nave espacial (WTF). Devuelve un entero:
  - 1 Si el primer operando es mayor que el segundo.
  - 0 Si el primer operando es igual que el segundo.
  - -1 Si el primer operando es menor que el segundo.

Operador Elvis (?:)

Devuelve un valor en caso de ser afirmativo, en otro caso, devolverá otro que se le especifique. `var1 ?: var2` es un equivalente a `var1 ? var1 : var2`.

Tabla de comparaciones

Comparando arrays

Lo que se compara es la cantidad de elemtnos que tiene, independientemente del contenido. Si una clave del operando 1 no se encuentran en el operando 2 entonces los arrays son incomparables.

Estructuras de control