LẬP TRÌNH PHP
(Phần 2)

Hệ thống bài cũ

- Viết câu lệnh điều khiển
  - Viết mã cho biểu thức điều kiện
  - Viết cấu trúc lựa chọn
- Khởi tạo và sử dụng hàm
  - Các kỹ năng cơ bản để làm việc với hàm
  - Khởi tạo và sử dụng thư viện của hàm
- Khởi tạo và sử dụng đối tượng
  - Khởi tạo và sử dụng lớp
  - Viết hằng, thuộc tính và phương thức của lớp
  - Một số kỹ năng bổ sung
  - Làm việc với kế thừa

Nội dung bài học

1. Làm việc với chuỗi và số
2. Làm việc với ngày tháng
3. Làm việc với mảng

1. Làm việc với chuỗi và số

Trong phần này có các nội dung:

1.1. Làm việc với chuỗi

1.2. Làm việc với số

1.3. Các kỹ năng khác

1.1. Làm việc với chuỗi

Trong phần này có các nội dung:

1.1.1. Tạo chuỗi

1.1.2. Thêm các ký tự đặc biệt vào chuỗi

1.1.3. Làm việc với độ dài chuỗi và chuỗi con

1.1.4. Tìm kiếm trong chuỗi

1.1.5. Thay thế một bộ phận trong chuỗi

1.1.6. Sửa chuỗi

1.1.7. Chuyển đổi giữa chuỗi và mảng

1.1.1. Tạo chuỗi

- Cách 1: gán chuỗi với dấu nháy đơn ‘’

```
$language = 'PHP';
```

- Cách 2: gán chuỗi với dấu nháy kép “”. Với cách này ta có thể gán giá trị của các biến vào trong chuỗi dễ dàng (phép thay thế biến)

```
$language = "PHP";
$count = 12;
$item = "flower";
$message = "You bought $count $item";
```

- Cách 3: gán chuỗi bằng HereDoc

```
$message = <<<MESSAGE
Line 1
Line 2
MESSAGE;
```

Tạo chuỗi

- Cách 4: gán chuỗi bằng NowDoc

```
$message = <<<'MESSAGE'
Line 1
Line 2
MESSAGE;
```

- So sánh giữa các cách tạo chuỗi:

  Khi tạo chuỗi sử dụng dấu nháy kép “” hay cú pháp kiểu heredoc, PHP sẽ thực hiện phép thay thế biến. Khi cần, quá trình này sẽ chuyển đổi giá trị biến sang kiểu chuỗi

1.1.2. Thêm các ký tự đặc biệt vào chuỗi

- Sử dụng ký hiệu thoát nối tiếp

Thoát nối tiếp	Mô tả	Dùng cho
\\	\	Tất cả các chuỗi trừ nowdoc
\’	‘	Chuỗi sử dụng dấu nháy đơn ‘
\”	“	Chuỗi sử dụng dấu nháy kép “
\$	$	Chuỗi sử dụng “ và heredoc
\n	Dòng mới	Chuỗi sử dụng “ và heredoc
\t	Tab mới	Chuỗi sử dụng “ và heredoc
\v	Tab dọc	Chuỗi sử dụng “ và heredoc
\oo	Giá trị hệ bát phân	Chuỗi sử dụng “ và heredoc
\xhh	Giá trị hệ thập lục phân	Chuỗi sử dụng “ và heredoc

Thêm các ký tự đặc biệt vào chuỗi

- Sử dụng hàm htmlentities: Trả về chuỗi sau khi chuyển tất cả các ký tự HTML đặc biệt sang thực thể HTML

  Ví dụ:

```
//Không sử dụng hàm htmlentities
$copyright1 = "\xa9 2010";                  //kết quả là '© 2010'
echo $copyright1;                          //hiển thị 2010
//Sử dụng hàm htmlentities
$copyright2 = htmlentities("\xa9 2010");    //kết quả là '&copy; 2010'
echo $copyright2;                          //hiển thị © 2010
```

1.1.3. Làm việc với độ dài chuỗi và chuỗi con

- Kiểm tra chuỗi rỗng: sử dụng hàm empty($chuoi). Hàm này trả về TRUE nếu biến $chuoi là chuỗi rỗng (“”), có giá trị NULL hoặc không được thiết lập

Ví dụ:

```
If (empty($name)) {
  $message = 'You must enter the name.';
}
```

- Lấy độ dài chuỗi: dùng hàm strlen($str). Hàm này trả về độ dài của chuỗi

Ví dụ:

```
$name = 'Ray Harris';
$length = strlen($name);        //$length là 10
```

Làm việc với độ dài chuỗi và chuỗi con

- Trích ra chuỗi con từ chuỗi ban đầu: sử dụng hàm substr($str, $i[, $len]). Hàm này trả về chuỗi con của chuỗi $str bắt đầu từ vị trí được định bởi biến $i và chứa số ký tự được định bởi biến $len

  Ví dụ:

```
$name = 'Ray Harris';
$length = strlen($name);                    //$length là 10
$first_name = substr($name, 0, 3);          //$frst_name là 'Ray'
```

1.1.4. Tìm kiếm trong chuỗi

- Sử dụng hàm strpos($str1, $str2[,$offset]): Tìm kiếm $str2 trong $str1. Nếu $str2 được tìm thấy, trả về giá trị nguyên cho vị trí của $str2 trong $str1. Nếu không tìm thấy $str2, trả về FALSE
- Theo mặc định, quá trình tìm kiếm sẽ bắt đầu từ vị trí 0, tuy nhiên có thể sử dụng biến $offset để chỉ định vị trí bắt đầu

Ví dụ:

```
$name = 'Martin Van Buren';
$i = strpos($name,'');     //$i là 6
$i = strpos($name, '', 7); //$i là 10 - dùng offset để tìm khoảng trắng
                           //thứ hai
$i = strrpos($name, '');   //$i là 10 - tìm kiếm chuỗi theo chiều ngược
```

1.1.5. Thay thế một bộ phận trong chuỗi

- Sử dụng hàm str_replace($str1, $new, $str2): trả về chuỗi mới trong đó tất cả $str1 trong $str2 được thay bằng $new

Ví dụ:

```
$phone = '554.555.6624';
$phone = str_replace('.','-',$phone);   //$phone là '554-555-6624
```

1.1.6. Sửa chuỗi

- Sử dụng các hàm sửa chuỗi:

Tên hàm	Mô tả
ltrim($str)	Trả về chuỗi mới loại bỏ các khoảng trắng thừa bên trái chuỗi $str
rtrim($str)	Trả về chuỗi mới loại bỏ các khoảng trắng thừa bên phải chuỗi $str
trim($str)	Trả về chuỗi mới loại bỏ các khoảng trắng thừa 2 bên chuỗi $str
lcfrst($str)	Trả về chuỗi mới với ký tự đầu được viết thường
ucfrst($str)	Trả về chuỗi mới với ký tự đầu được viết hoa
ucwords($str)	Trả về chuỗi mới với chữ cái đầu của các từ được viết hoa
strtolower($str)	Trả về chuỗi mới với các chữ được viết thường
strtoupper($str)	Trả về chuỗi mới với các chữ được viết hoa

Sửa chuỗi

- Ví dụ:

```php
$name = '    ray harris    ';
$name = ltrim($name); //$name = 'ray harris    '
$name = rtrim($name); //$name = 'ray harris'
$name = strtolower($name); //$name = 'ray harris'
$name = strtoupper($name); //$name = 'RAY HARRIS'
```

1.1.7. Chuyển đổi giữa chuỗi và mảng

- Sử dụng các hàm chuyển đổi:

Tên hàm	Mô tả
`explode($sep, $str)`	Trả về mảng các chuỗi con được phân cách bằng chuỗi $sep trong chuỗi cha $str. Chuỗi con này có thể có độ dài bất kỳ. Nếu nó là chuỗi rỗng thì mỗi chuỗi con trong mảng sẽ chứa một ký tự đơn.
`implode($sep, $sa)`	Trả về chuỗi được nối từ các phần tử trong mảng $sa và được phân cách bằng chuỗi $sep. Chuỗi $sep là bắt buộc nhưng có thể bằng rỗng.

- Ví dụ:

```
$names = 'Mike|Anne|Joel|Ray';
$names = explode('|', $name);
$name1 = $names[0]; // $name1 = 'Mike'
$name2 = $names[1]; // $name2 = 'Anne'
$names = implode('|', $names); // $names = 'Mike|Anne|Joel|Ray'
```

1.1.8. So sánh chuỗi

- Sử dụng các hàm so sánh:

Tên hàm	Mô tả
`strcmp($str1, $str2)`	So sánh hai chuỗi và trả về số nguyên chỉ thứ tự của chúng: -1 nếu $str1 đứng trước $str2, 1 nếu $str1 đứng sau $str2 và 0 nếu giống nhau. So sánh này phân biệt chữ hoa chữ thường. Vì vậy, chữ hoa đứng trước chữ thường.
`strcasecmp($str1, $str2)`	Phiên bản khác của strcasecmp nhưng không phân biệt chữ hoa chữ thường.
`strnatcmp($str1, $str2)`	Phiên bản khác của strcasecamp nhưng sử dụng so sánh tự nhiên cho các số trong chuỗi.
`strnatcasecmp($str1, $str2)`	Phiên bản khác của strcmp nhưng sử dụng so sánh tự nhiên và không phân biệt chữ hoa chữ thường.

- Ví dụ:

```
$result = strcmp('Anders', 'Zylka'); //$result = -1 (A trước Z)
$result = strcmp('Anders', 'zylka'); //$result = 1 (z đứng trước A)
$result = strcmp('img6', 'img10');  //$result=1 (vì img10 có số 1 đứng
                                    //trước số 6)
```

1.2. Làm việc với số

Trong phần này có các nội dung:

1.2.1. Sử dụng các hàm toán học

1.2.2. Sinh số ngẫu nhiên

1.2.1. Sử dụng các hàm toán học

- Một số hàm toán học thông dụng:

Tên hàm	Mô tả
`abs($value)`	Trả về trị tuyệt đối của số $value.
`ceil($value)`	Trả về giá trị được làm tròn lên tròn số gần nhất.
`floor($value)`	Trả về giá trị được làm tròn xuống tròn số gần nhất.
`max($n1, $n2[, $n3 …]`	Trả về giá trị lớn nhất của dãy số truyền vào.
`min($n1, $n2[, $n3 …]`	Trả về giá trị nhỏ nhất của dãy số truyền vào.
`Pi`	Trả về giá trị PI (xấp xỉ 3.141593).
`pow($base, $exp)`	Trả về giá trị mũ $exp của số nguyên $base. Cả hai tham số truyền vào đều có thể là số chấm động.
`round($value[, $precision])`	Trả về giá trị tròn số của $value tới số chấm động có số chữ số thấp phân được xác định là $precision. Nếu $precision bị bỏ qua, giá trị mặc định là 0. Nếu $precision âm, hàm làm tròn về tròn số. Ví dụ -1 làm tròn tới số hàng chục gần nhất còn -2 làm tròn số hàng trăm gần nhất.
`sqrt($value)`	Trả về căn bậc hai của $value.

Sử dụng các hàm toán học

- Ví dụ:

```php
$subtotal = 15.99;
$tax_rate = 0.08;
$tax = round($subtotal * $tax_rate, 2); //1.28 thay vì 1.2792
$num1 = 4;
$root =sqrt($num1); //2
$num2 = 5;
$power = pow($num2, 2); // 5 mũ 2 = 25
```

1.2.2. Sinh số ngẫu nhiên

- Các hàm sinh số ngẫu nhiên:

Tên hàm	Mô tả
`getrandmax()`	Trả về số nguyên lớn nhất mà hàm rand có thể lấy. Nó có thể là số tương đối nhỏ (32000) trên một số hệ thống.
`rand()`	Trả về số nguyên ngẫu nhiên trong khoảng từ 0 đến số lớn nhất trong hàm getrandmax().
`rand($lo, $hi)`	Trả về số nguyên ngẫu nhiên trong khoảng giữa $lo và $hi.
`mt_getrandmax()`	Trả về số nguyên lớn nhất mà hàm mt_rand có thể trả về. Số này thường là PHP_INT_MAX.
`mt_rand()`	Trả về số nguyên ngẫu nhiên giữa 0 và mt_getrandmax(). Hàm này sử dụng thuật toán Mersen Twister chạy nhanh hơn và "ngẫu nhiên hơn" thuật toán mà hàm rand sử dụng.
`mt_rand($lo, $hi)`	Trả về số ngẫu nhiên giữa khoảng $lo và $hi sử dụng thuật toán Mersene Twister.

Sinh số ngẫu nhiên

- Ví dụ:

```php
//Sinh mật khẩu ngẫu nhiên
$password_length = 8;

//Thêm một ký hiệu vào mật khẩu
$symbols = '~!@#$%^&*()-_=+[]{};:,.<>?';  //chuỗi ký hiệu sẽ lấy ra
$symbol_count = strlen($symbols);
$index = mt_rand(0, $symbol_count - 1);   //chọn một vị trí ngẫu nhiên
$password = substr($symbols, $index, 1);  //thêm một ký hiệu
$password .= chr(mt_rand(48, 57));        //thêm một số
$password .= chr(mt_rand(65, 90));        //thêm một chữ hoa

//Thêm các chữ thường cho tới khi đạt tới độ dài cần thiết
while (strlen($password) < $password_length) {
  $password .= chr(mt_rand(97, 122));
}
$password = str_shuffe($password);        //tráo ngẫu nhiên các ký tự
echo $password; //ví dụ 'd7kug-Hf'
```

1.3. Các kỹ năng khác

Trong phần này có các nội dung:

1.3.1. Định dạng lại chuỗi và số

1.3.2. Chuyển chuỗi thành số

1.3.1. Định dạng lại chuỗi và số

- Sử dụng hàm sprintf($format, $var1[, $var2...]): Trả về chuỗi chứa một hoặc nhiều giá trị được định dạng theo tham số $format
- Mã định dạng luôn bắt đầu bằng ký hiệu % và kết thúc là ký tự xác định kiểu dữ liệu
- Bảng mã các kiểu chuyển dữ liệu:

Ký tự	Mô tả
s (string)	Định dạng giá trị thành kiểu chuỗi.
d (digit)	Định dạng giá trị kiểu số nguyên.
f (float)	Định dạng giá trị kiểu số chấm động.
e (exponent)	Định dạng giá trị theo cách viết số mũ.
c (char)	Định dạng ký tự tương ứng với mã ASCII.
o (binary)	Định dạng số nguyên theo hệ nhị phân.
o (octal)	Định dạng số nguyên theo hệ bát phân.
x (hexa)	Định dạng số nguyên theo hệ thập lục phân (chữ thường).
X	Định dạng số nguyên theo hệ thập lục phân (chữ hoa).

Định dạng lại chuỗi và số

- Ví dụ:

```php
$s1 = sprintf('It cost %s dollars', 12); //It cost 12 dollars
$s2 = sprintf('%s', 4.5); //4.5
$s3 = sprintf('%s', 9451000.000000); //9451000
$s4 = sprintf('%f', 9.451e6); //9451000.00000
$s5 = sprintf('%e', 9451000.00000); //9.451000e+6
$s6 = sprintf('%c', 65); //A
$s7 = sprintf('%x', 15); //f
$s8 = sprintf('%X', 15); //F
$s9 = sprintf('%s%%', 4.5); //4.5%
```

1.3.2. Chuyển chuỗi thành số

- Cách 1: Viết kiểu cần ép trong cặp ngoặc đơn, theo sau là giá trị cần ép kiểu
- Cách 2: Sử dụng các hàm chuyển đổi:

Hàm	Mô tả
`intval($var)`	Trả về giá trị nguyên cho tham số $var.
`floatval($var)`	Trả về giá trị chấm động cho tham số $var.

- Ví dụ:

```
$value_3 = (int) '42 miles'; //kết quả là 42
$value_4 = (int) '2,500 feet'; //kết quả là 2
$value = intval('42'); //kết quả là 42
$value_1 = (float) '4.2'; //kết quả là 4.2
$value_2 = (float) '4.2 gallons'; //kết quả là 4.2
$value = floatval('4.2') //kết quả là 4.2
```

2. Làm việc với ngày tháng

Trong phần này có các nội dung:

2.1. Sử dụng nhãn thời gian

2.2. Sử dụng đối tượng

2.1. Sử dụng nhãn thời gian

- Nhãn thời gian sử dụng số nguyên để biểu thị ngày tháng và thời gian. Số nguyên này lưu số giây tính từ nửa đêm ngày 1-1-1970 theo giờ GMT (Greenwich Mean Time – giờ quốc tế Greenwich)
- Tạo và định dạng nhãn thời gian:
  - Sử dụng hàm date($format, [$ts]: Trả về chuỗi mô tả ngày tháng đã được định dạng theo chuỗi địnhdạng $format. Theo mặc định, hàm làm việc với ngày giờ hiện tại. Tuy nhiên có thể dùng tham số $ts để chỉ định nhãn thời gian cho bất kỳ ngày giờ nào
  - Các mã định dạng thông dụng của hàm date:

Sử dụng nhãn thời gian

Ký tự	Mô tả	Từ	Tới
D	Ngày trong tuần – 3 chữ cái	Mon	Sun
l (chữ thường)	Ngày trong tuần – đủ chữ	Monday	Sunday
n	Tháng – không có số 0 ở đầu	1	12
m	Tháng – có 0 ở đầu	01	12
M	Tháng – 3 chữ cái đầu	Jan	Dec
F	Tháng – đủ chữ	January	December
j	Ngày trong tháng – không có số 0 ở đầu	01	31
d	Ngày trong tháng – có 0 ở đầu	01	31
Y	Năm – 4 chữ số	2010	
L	Năm nhuận (1) hoặc năm thường (0)	0	1
g	Giờ - định dạng 12-giờ, không có 0 ở đầu	1	12
h	Giờ - định dạng 24-giờ, không có 0 ở đầu	0	23
G	Giờ - định dạng 12-giờ, có 0 ở đầu	01	12
H	Giờ - định dạng 24-giờ, có 0 ở đầu	00	23
i	Phút – có 0 ở đầu	00	59
s	Giây – có 0 ở đầu	00	59
a	am/pm – viết thường	am	pm
A	AM/PM – viết hoa	AM	PM
T	Viết tắt cho vùng thời gian	EST	
U	Số giây kể từ Kỷ nguyên Unix	-2,147,483,648	2,147,483,647

Sử dụng nhãn thời gian

- Ví dụ:

```php
$date1 = date('n/j/Y');                 //3/15/2011
$date2 = date('Y-m-d');                 //2011-03-15
$date3 = date('l, F d, Y');             //Monday, March 15, 2011
$date4 = date('g:i a');                 //1:30 pm
$date5 = date('H:i:s');                 //13:30:00
$date6 = date('H:i:s \a\t H:i:s');      //2011-03-15 at 13:30:00
$date7 = date('Y-m-d', 1331843400);     //2012-03-15
```

Sử dụng nhãn thời gian

- Làm việc với nhãn thời gian:
  - Sử dụng các hàm làm việc:

Tên hàm	Mô tả
`time()`	Trả về ngày giờ hiện tại dưới dạng nhãn thời gian.
`mktime([$h[, $m[, $s[, $M[, $D[, $Y]]]]]])`	Trả về nhãn thời gian dựa vào ngày giờ được truyền vào. Nếu có bất cứ thành phần nào bị bỏ qua, hàm sẽ lấy giá trị của ngày giờ hiện tại.
`checkdate($M, $D, $Y)`	Trả về TRUE nếu ngày, tháng, năm là hợp lệ.
`getdate([$ts])`	Nếu tham số bị bỏ qua, hàm lấy ngày giờ hiện tại.

- Ví dụ:

```
$now = time();                              //ví dụ 1265656521
$expires = mktime(13, 30, 0, 3, 15, 2012);//15/3/2012 13:30:00
$expires = mktime(13, 30, 0, 3, 15);      //sử dụng năm hiện tại
$expires = mktime(13, 30, 0, 3, 15);      //sử dụng năm và ngày hiện tại
```

Sử dụng nhãn thời gian

- Hướng dẫn lấy các thành phần của nhãn thời gian:

```
$expires = mktime(13, 30, 0, 3, 15, 2012);
$parts = getdate($expires);
$year = $parts['year'];                    //2012
$mon = $parts['mon'];                     //dạng số, 3
$month = $parts['month'];                  //dạng chữ, 'March'
$mday = $parts['mday'];                    //ngày trong tháng, 15
$weekday = $parts['weekday'];               //thứ trong tuần, 'Monday'
$wday = $parts['wday'];                    //thứ trong tuần dạng số, 1
$hours = $parts['hours'];                  //giờ, 13
$minutes = $parts['minutes'];               //phút, 30
$seconds = $parts['seconds'];               //giây, 0
```

Sử dụng nhãn thời gian

- Cách khác để tạo và làm việc với nhãn thời gian:
  - Sử dụng hàm strtotime($str[, $ts]): Trả về nhãn thời gian cho chuỗi truyền vào. Theo mặc định, hàm này sẽ làm việc tương đối với ngày giờ hiện tại
  - Các kiểu mẫu dùng cho hàm strtotime:

Kiểu	Mô tả
Tuyệt đối	Là chuỗi định dạng ngày tháng và thời gian. Nếu không có thông tin thời gian, thời gian sẽ được thiết lập mặc định là nửa đêm. Khi một số thành phần thời gian (ví dụ, chỉ có tháng và ngày) được truyền vào thì các bộ phận khác mặc định không đổi. Nếu không có thông tin về ngày tháng, ngày tháng sẽ lấy theo ngày hiện tại. Một số thành phần của thời gian (ví dụ chỉ có giờ và phút) không được chỉ ra sẽ được thiết lập về 0.
Tương đối	Khoảng cách được thêm vào hoặc trừ bớt đi từ ngày giờ gốc. Một số ngày tương đối thiết lập giờ về thời điểm nửa đêm.

Sử dụng nhãn thời gian

- Ví dụ:

```
//Hướng dẫn tạo nhãn thời gian với mẫu tuyệt đối
//giả định ngày giờ hiện tại là Chủ nhật ngày 08/04/2012 1:30:00 chiều
$date1 = strtotime('2013-06-01');          //thứ bảy 01/06/2013 00:00
$date3 = strtotime('Jun 1');              //thứ sáu 01/06/2012 00:00
$date6 = strtotime('2013-02-29 8:45am');    //thứ Sáu 01/03/2013 08:45

//Hướng dẫn tạo nhãn thời gian với mẫu tương đối
$date1 = strtotime('+1 hour');           //Chủ nhật 08/04/2012 14:30
$date4 = strtotime('tomorrow 10:15am');   //thứ hai 09/04/2012 10:15
$date9 = strtotime('now second tue of 8am');   //thứ ba 13/11/2012 08:00

//Hướng dẫn thay đổi nhãn thời gian
$checkout = mktime(13, 30, 0, 4, 8, 2012);
$due_date = strtotime('+3 weeks 6pm', $checkout);
```

2.2. Sử dụng đối tượng

- Lớp DateTime cung cấp phương thức hướng đối tượng để làm việc với ngày tháng và thời gian.
- Sử dụng đối tượng DateTime:
  - Tạo đối tượng DateTime:

```php
//Sử dụng ngày giờ hiện tại
$now = new DateTime();
//Sử dụng chuỗi định dạng của strtotime
$expires = new DateTime('2012-03-15 13:30:00');
$tomorrow = new DateTime('+1 day');
```

  - Các hàm của đối tượng DateTime:

Sử dụng đối tượng

Tên	Mô tả
`format($format)`	Trả về chuỗi với ngày giờ được định dạng như định nghĩa trong chuỗi định dạng. Hàm này sử dụng đúng các mã định dạng như hàm date.
`setTime($h, $m, $s)`	Thiết lập giờ.
`setDate($y, $m, $d)`	Thiết lập ngày tháng.
`modify($str)`	Thay đổi ngày giờ dựa vào mã định dạng. Hàm hoạt động giống hàm strtotime với mẫu tương đối.
`getTimestamp()`	Lấy ngày giờ hiện tại làm nhãn thời gian.
`setTimestamp($str)`	Thiết lập ngày giờ dựa vào nhãn thời gian.

Sử dụng đối tượng

Ví dụ:

```php
//Hướng dẫn sao chép đối tượng DateTime
$invoice_date = new DateTime('2012-03-15 13:30:00');
$due_date = clone $invoice_date;

//Hướng dẫn thiết lập ngày giờ của đối tượng DateTime
$due_date->setTime(22, 30, 0); //10:30 chiều
$due_date->setDate(2012, 3, 15); //ngày 15/3/2012

//Hướng dẫn thay đổi đối tượng DateTime
$due_date->modify('+3 weeks');

//Hướng dẫn hiển thị đối tượng DateTime
echo 'Payment Due: ' . $due_date->format('M. j, Y \a\t g:i a');

//Hướng dẫn hoán chuyển nhãn thời gian thành đối tượng DateTime
$tomorrow = strtotime('tomorrow 8am');
$nextday = new DateTime();
$nextday->setTimestamp($tomorrow);
```

Sử dụng đối tượng

- Sử dụng đối tượng DateInterval (khoảng ngày tháng):
  - Tạo đối tượng DateInterval:

```php
$interval = new DateInterval('P30D'); //30 ngày
```

  - Các bộ phận của chuỗi khoảng:

Bộ phận	Mô tả	Ví dụ
P	Bắt đầu mã khoảng thời gian.	
nY	Định nghĩa số năm.	'P1Y'
nM	Định nghĩa số tháng	'P2M', 'P1Y6M'
nW	Định nghĩa số tuần	'P3W', 'P1M2W'
nD	Định nghĩa số ngày	'P4D', 'P1Y6M4D'
T	Bắt đầu phần thời gian của mã	
nH	Định nghĩa số giờ	'PT3H', 'P3DT12H'
nM	Định nghĩa số phút	'PT10M', 'P1MT4H30M'
nS	Định nghĩa số giây	'PT30S', 'P1Y1MT1M10S'

Sử dụng đối tượng

- Ví dụ: Hướng dẫn sử dụng chuỗi khoảng thời gian

```
$interval_1 = new DateInterval('P1Y2M10D'); //1 năm, 2 tháng, 20 ngày
$interval_2 = new DateInterval('PT1H2M3S'); //1 giờ, 2 phút, 3 giây
$interval_3 = new DateInterval('P1Y2M3DT1H2M3S');
```

- Hàm format của đối tượng DateInterval:

Tên	Mô tả
`format($format)`	Trả về chuỗi được định dạng theo chuỗi định nghĩa định dạng truyền vào.

Sử dụng đối tượng

- Mã định dạng của hàm format:

Mã	Mô tả
%R	Dấu của khoảng thời gian, + cho dương và – cho âm
%y	Năm
%m	Tháng
%d	Ngày
%h	Giờ
%i	Phút
%s	Giây

- Ví dụ:

```
//Hướng dẫn hiển thị khoảng ngày tháng
echo $interval_1->format('%m months, %d days');      //'2 months, 10 days'
echo $interval_1->format('%R %M months');          //'+ 02 months'
echo $interval_1->format('%R %y %m %d %h %i %s');  //'+ 1 2 10 0 0 0'
echo $interval_1->format('%R%yy %mm %đ %H:%I:%S'); //'+1y 2m 10d 00:00:00'
```

Sử dụng đối tượng

- Phối hợp đối tượng DateTime và DateInterval:
  - Sử dụng các phương thức:

Tên	Mô tả
`add($interval)`	Thêm lượng thời gian được xác định bằng đối tượng DateInterval truyền vào.
`sub($interval)`	Trừ lượng thời gian mà đối tượng $interval truyền vào.
`diff($date)`	Trả về đối tượng DateInterval biểu thị lượng thời gian giữa đối tượng DateTime hiện tại với đối tượng DateTime truyền vào. Nếu đối tượng DateTime truyền vào nhỏ hơn thì hàm trả về đối tượng Interval âm.

  - Ví dụ:

```
//Hướng dẫn cộng một đối tượng DateInterval vào đối tượng DateTime
$checkout_length = new DateInterval('P3W');
$due_date = new DateTime();
$due_date->add($checkout_length);
```

3. Làm việc với mảng

Trong phần này có các nội dung:

3.1. Khởi tạo và sử dụng mảng

3.2. Khởi tạo và sử dụng mảng liên kết

3.3. Làm việc với hàng đợi và ngăn xếp

3.4. Làm việc với mảng hai chiều (mảng của mảng)

3.1. Khởi tạo và sử dụng mảng

- Mảng chứa một hoặc nhiều phần tử. Mỗi phần tử của mảng chứa một chỉ mục (index) và một giá trị (value). Chỉ mục có thể là số tự nhiên hoặc chuỗi, còn giá trị có thể là kiểu dữ liệu PHP bất kỳ.
- Chỉ mục số tự nhiên với số 0 được đánh cho phần tử thứ nhất, số 1 cho phần tử thứ 2 và tương tự
- Cú pháp tạo mảng:

```
$array_name = array([value1 [, value 2, ...]])
```

- Cú pháp tham chiếu tới phần tử mảng: `$array_name[index]`
- Ví dụ tạo mảng:

```
//Bằng một câu lệnh
$names = array('Ted Lewis', 'Sue Jones', 'Ray Thomas');

//Bằng nhiều câu lệnh
$names = array();   //tạo mảng rỗng
$names[0] = 'Ted Lewis'; //gán ba giá trị vào mảng
$names[1] = 'Sue Jones';
$names[2] = 'Ray Thomas';
```

Khởi tạo và sử dụng mảng

- Cú pháp thêm phần tử vào cuối mảng:

```
$array_name[] = $value;
```

- Hàm xóa giá trị khỏi phần tử trong mảng:

Tên hàm	Mô tả
`unset($var1, [, var2 ...]`	Xóa giá trị trong phần tử mảng hoặc xóa toàn bộ mảng bằng cách gán cho nó giá trị bằng NULL.
`array_values($array)`	Trả về mảng mới chứa tất cả các giá trị của mảng cũ sau khi các giá trị NULL bị loại bỏ và chỉ mục được đánh lại.

- Hướng dẫn thêm một giá trị vào cuối mảng:

```
$letters = array('a', 'b', 'c', 'd');      //mảng chứa bốn chữ a, b, c, d
$letters[] = 'e';                          //mảng chứa năm chữ a,b,c,d,e
```

3.2. Khởi tạo và sử dụng mảng liên kết

Trong phần này có các nội dung:

3.2.1. Tạo mảng liên kết

3.2.2. Thêm và xóa phần tử của mảng liên kết

3.2.3. Sử dụng vòng lặp foreach để làm việc với mảng liên kết

3.2.1. Tạo mảng liên kết

- Mảng liên kết: sử dụng chỉ mục kiểu chuỗi cho giá trị được lưu trong mảng
- Cú pháp tạo mảng liên kết:

```
array([key1 => value1, key2 => value2, ...])
```

- Mảng liên kết cũng có thể được tạo bằng nhiều câu lệnh

Ví dụ:

```
//Tạo mảng liên kết bằng một câu lệnh
$tax_rates = array('NC' => 7.75,'CA' => 8.25, 'NY' => 8.875);

//Tạo mảng liên kết bằng nhiều câu lệnh
$tax_rates = array();
$tax_rates['NC'] = 7.75;
$tax_rates['CA'] = 8.25;
$tax_rates['NY'] = 8.875;
```

3.2.2. Thêm và xóa phần tử của mảng liên kết

```php
//Khởi tạo mảng
$name = array('first' => 'Ray', 'last' => 'Harris');

//Thêm một phần tử
$name['middle'] = 'Thomas';

//Lấy giá trị của một khóa cụ thể
$name = array('first' => 'Ray', 'last' => 'Harris');
$first_name = $name['first'];

//Xóa phần tử của mảng
unset($name['first']);  //xóa giá trị của phần tử có key là first
unset($name);           //xóa toàn bộ các phần tử và $name = NULL
```

3.2.3. Sử dụng vòng lặp foreach để làm việc với mảng

- Cú pháp:

```
foreach ($array_name as [ $key => ] $value) {
//các câu lệnh sử dụng hai biến $key và $value
}
```

- Ví dụ:

```
$tax_rates = array('NC' => 7.75,'CA' => 8.25, 'NY' => 8.875);
//Vòng lặp foreach hiển thị giá trị trong mảng liên kết
foreach ($tax_rates as $rate) {
    echo $rate; //Kết quả: 7.75 8.258.875
}
//Vòng lặp foreach hiển thị khóa và giá trị
foreach ($tax_rates as $state => $rate) {
    echo "$state = $rate"; //Kết quả: NC=7.75 CA=8.25 NY=8.875
}
```

3.3. Làm việc với hàng đợi và ngăn xếp

- Các hàm làm việc với hàng đợi và ngăn xếp:

Tên hàm	Mô tả
`array_push($array, $value)`	Thêm giá trị $value vào cuối chuỗi $array.
`array_pop($array)`	Loại bỏ và trả về giá trị cuối cùng trong chuỗi $array.
`array_unshift($array, $value)`	Thêm giá trị $value vào đầu chuỗi $array.
`array_shift($array)`	Loại bỏ và trả về giá trị đầu tiên của chuỗi $array.

- Ví dụ:

```php
//Làm việc với ngăn xếp
$names = array('Mike', 'Joel', 'Anne');
array_push($names, 'Ray'); //mảng kết quả: Mike, Joel, Anne, Ray
$next = array_pop($names); //mảng kết quả: Mike, Joel, Anne
echo $next; //Ray

//Làm việc với hàng đợi
$names = array('Mike', 'Joel', 'Anne');
array_push($names, 'Ray'); //mảng kết quả: Mike, Joel, Anne, Ray
$next = array_shift($names); //mảng kết quả: Anne, Joel, Ray
echo $next; //Mike
```

3.4. Làm việc với mảng hai chiều (mảng của mảng)

- Mảng hai chiều: mỗi phần tử của mảng là một mảng
- Hướng dẫn khởi tạo và sử dụng mảng hai chiều:

```
//Khởi tạo mảng 2 chiều rỗng
$cart = array();

//Tạo và thêm mảng 1 chiều vào mảng 2 chiều
$item = array(); //Khởi tạo mảng 1 chiều rỗng
$item['itemCode'] = 123;
$item['itemName'] = 'Visual Basic 2010';
$item['itemCost'] = 52.5;
$item['itemQuantity'] = 5;
$cart[]= $item; //Thêm mảng vừa tạo vào mảng 2 chiều

//Tạo và thêm mảng 1 chiều khác vào mảng 2 chiều
$item = array(); //Khởi tạo mảng 1 chiều rỗng
$item['itemCode'] = 456;
$item['itemName'] = 'C++ 2010';
$cart[]= $item; //Thêm mảng vừa tạo vào mảng 2 chiều

//Tham chiếu tới các phần tử trong mảng 2 chiều
echo $cart[0]['itemCode']; // 123
echo $cart[1]['itemName']; //C++ 2010
```

Tổng kết bài học

- Có thể tạo ra chuỗi với phép thay thế chuỗi, cú pháp heredoc và cú pháp nowdoc
- Thêm các ký tự đặc biệt vào chuỗi bằng cách dùng thoát nối tiếp
- Hàm làm việc với chuỗi: lấy độ dài của chuỗi, tìm kiếm và thay thế một chuỗi con, rút gọn, lấy đệm và thay thế chuỗi, chuyển đổi chuỗi sang mảng, so sánh hai chuỗi và định dạng chuỗi
- Sử dụng phép chuyển đổi kiểu hoặc hàm intval/floatval để chuyển một chuỗi sang số nguyên hoặc số có dấu chấm động

Tổng kết bài học

- Nhãn thời gian là một số nguyên biểu thị ngày giờ và được tính bằng số giây kể từ nửa đêm ngày 1/1/1970 (theo giờ GMT)
- Lớp DateTime cung cấp phương thức lập trình hướng đối tượng để làm việc với ngày giờ
- Đối tượng DateInterval làm việc với độ dài khoảng thời gian
- Mảng trong PHP có tính động
- Mảng liên kết là mảng mà chỉ mục có giá trị kiểu chuỗi
- Vòng lặp for được dùng nhiều để duyệt mảng có chỉ mục kiểu số nguyên, vòng lặp foreach sử dụng thông dụng để duyệt các mảng liên kết
- Mảng hai chiều (mảng của mảng), là mảng trong đó các phần tử lại chứa một mảng khác

XIN CẢM ƠN!