Bài 5

PHP cơ bản

Viện CNTT & TT

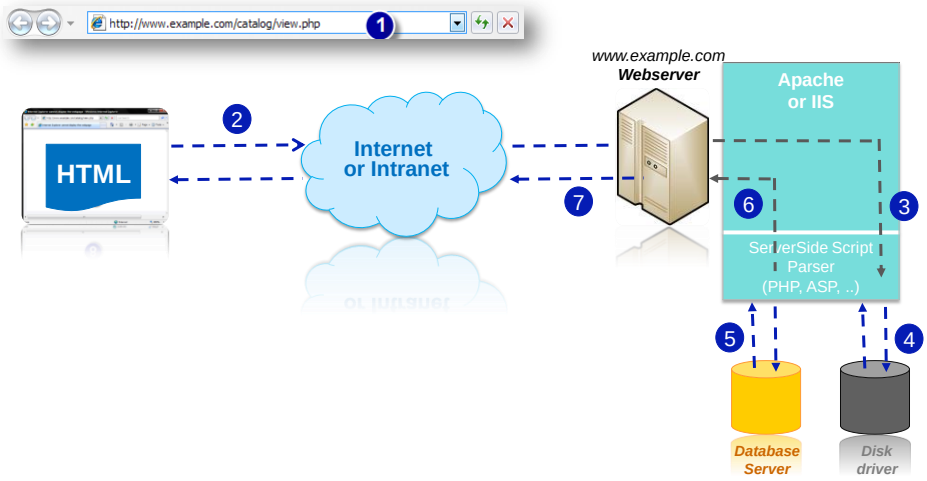
1. Giới thiệu về PHP

- PHP là gì?
  - PHP = **P**HP **H**ypertext **P**reprocessor, tên gốc là Personal Home Pages.
  - PHP là ngôn ngữ viết web động.
  - Bộ biên dịch PHP là phần mềm mã nguồn mở.
  - Là ngôn ngữ server-side script, tương tự như ASP, JSP, … thực thi ở phía WebServer
  - Thường kết nối với hệ quản trị CSDL MySQL

Giới thiệu về PHP – Lịch sử phát triển

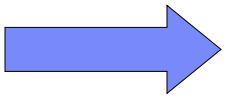
- **PHP :** Rasmus Lerdorf in 1994 (được phát triển để phát sinh các form đăng nhập sử dụng giao thức HTTP của Unix)
- **PHP 2 (1995) :** Chuyển sang ngôn ngữ script xử lý trên server. Hỗ trợ CSDL, Upload File, khai báo biến, mảng, hàm đệ quy, câu điều kiện, biểu thức, …
- **PHP 3 (1998) :** Hỗ trợ ODBC, đa hệ điều hành, giao thức email (SNMP, IMAP), bộ phân tích mã PHP (parser) của Zeev Suraski và Andi Gutmans
- **PHP 4 (2000) :** Trợ thành một thành phần độc lập cho các webserver. Parse đổi tên thành Zend Engine. Bổ sung các tính năng bảo mật cho PHP
- **PHP 5 (2005) :** Bổ sung Zend Engine II hỗ trợ lập trình HĐT, XML, SOAP cho Web Services, SQLite

Cơ chế hoạt động của WebServer



Cài đặt

- **Để thiết kế trang web sử dụng PHP & MySQL, cần cài đặt:**
  - Máy chủ web Apache
  - PHP
  - Hệ quản trị cơ sở dữ liệu MySQL



XAMPP

Một số khái niệm

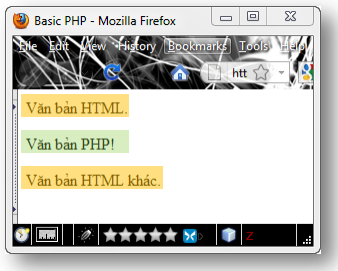
- PHP nhúng vào HTML
  - Có thể nhúng mã PHP vào mọi vị trí trong tài liệu HTML.
  - Chèn mã PHP vào file HTML: Có 3 dạng chính

```
<?php echo("Hello World!"); ?>

<? echo("Hello World!"); ?>

<script language="php">
        echo("Hello World!");
</script>
```

  - Phần mở rộng của tập tin chứa mã PHP thường là **.php:** index.php, giohang.php, ...

Ví dụ 1

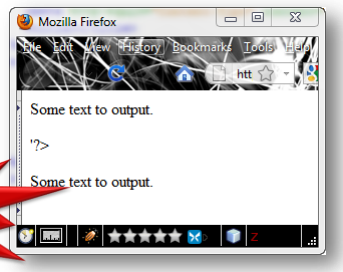
```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <html>
3      <head>
4          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5          <title>Basic PHP</title>
6      </head>
7      <body>
8          <p>Văn bản HTML.</p>
9          <?php
10             echo '<p>Văn bản PHP!</p>';
11         ?>
12         <p>Văn bản HTML khác.</p>
13     </body>
14 </html>
```



http://localhost/01BasicPhp/01Quytoc_VD3.php

Ví dụ 3

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <html>
3      <head>
4          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5          <title></title>
6      </head>
7      <body>
8          <?='<p>Some text to output.</p>'?>
9
10         <?php
11             echo '<p>Some text to output.</p>';
12         ?>
13     </body>
14 </html>
```



Không nên sử dụng cú pháp PHP viết tắt

Chỉnh sửa php.ini

Tham số	Ý nghĩa
short_open_tag = Off	Cho phép sử dụng <? ?>
asp_tags = Off	Cho phép sử dụng <% %>
post_max_size = 8M	Kích thước tối đa của dữ liệu gửi lên server
file_uploads = On	Cho phép upload file
upload_max_filesize = 2M	Kích thước tối đa của mỗi file upload

Một số khái niệm

- Đặc điểm của PHP
  - Có khả năng đối tượng
  - Thông dịch
  - Phân biệt chữ hoa và chữ thường
  - Lệnh kết thúc bằng dấu chấm phẩy " ; "
  - PHP là một ngôn ngữ kịch bản ràng buộc lỏng:
    - Không cần khai báo trước, việc khai báo sẽ được tự động thực hiện khi sử dụng.
    - Không cần định kiểu. Kiểu giá trị sẽ được xác định phù hợp với dữ liệu đầu vào

Một số khái niệm

- Tại sao sử dụng PHP?
  - PHP dễ học, dễ viết.
  - Có khả năng truy xuất hầu hết CSDL có sẵn.
  - Thể hiện được tính bền vững, chặn chẽ, phát triển không giới hạn.
  - PHP miễn phí, mã nguồn mở.

Viết ghi chú trong PHP

Để ghi chú trong PHP có 3 dạng sau:

Dạng 1: # đây là ghi chú.

Dạng này chỉ áp dụng ghi đó chỉ nằm trên một dòng văn bản

Dạng 2: // đây là ghi chú.

Dạng này cũng chỉ áp dụng ghi đó chỉ nằm trên một dòng văn bản

Dạng 3: /* đây là một ghi chú dài

Áp dụng cho nhiều hàng */

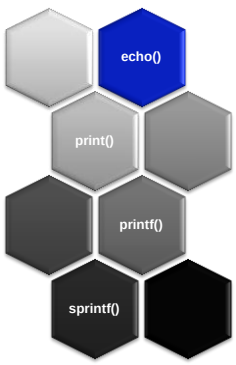
Khai báo và gán giá trị cho biến

- Khai báo biến
  - Cú pháp: $tên_biến
  - Ví dụ: $tong
- Quy tắc đặt tên cho biến
  - Tên biến phải bắt đầu bằng ký tự $, theo sau là 1 ký tự hoặc dấu _, tiếp đó là ký tự, ký số hoặc dấu _
  - Nên khởi tạo giá trị ban đầu cho biến
  - Tên biến không trùng với tên hàm
  - Biến không nên bắt đầu bằng ký số
  - Tên biến có phân biệt chữ HOA – chữ thường

Phạm vi hoạt động của biến

- Biến cục bộ
  - Biến được khai báo trong hàm => biến cục bộ
  - Khi ra khỏi hàm => biến cục bộ và giá trị của nó sẽ bị hủy bỏ
  - Lưu ý: khi có cùng tên thì biến bên trong hàm và biến bên ngoài hàm là hai biến hoàn toàn khác nhau
- Biến toàn cục
  - Có thể truy xuất bất cứ nơi nào trong trang
  - Khi muốn sử dụng và cập nhật biến toàn cục trong hàm thì phải dùng từ khóa global phía trước biến hoặc dùng $_GLOBALS[“tên_biến”]
- Biến static
  - Không mất đi giá trị khi ra khỏi hàm
  - Sẽ giữ nguyên giá trị trước đó khi hàm được gọi một lần nữa
  - Phía trước tên biến static phải có từ khóa static

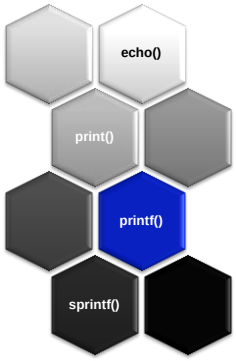
Xuất dữ liệu ra trình duyệt



- **Cú pháp:**

```
void echo(tham số chuỗi [, tham số chuỗi [, tham số chuỗi]])

int print(tham số)

boolean printf(string format [, mixed args])

string sprint(string format [, mixed args])
```

Xuất dữ liệu ra trình duyệt



Ký hiệu	Kiểu tham số	Thể hiện ra
%b	Số nguyên	Số nhị phân
%c	Số nguyên	1 ký tự ACSII
%d	Số nguyên	Số nguyên
%f	Số thực	Số thực
%o	Số nguyên	Số bát phân
%s	Chuỗi	Chuỗi
%u	Số nguyên	Số nguyên không dấu
%x	Số nguyên	Số thập lục phân thường
%X	Số nguyên	Số thập lục phân hoa

```
printf("$%.2f", 43.2);   // $43.20
```


Khai báo hằng

- Dùng hàm define() để định nghĩa. Một khi hằng được định nghĩa, nó không bị thay đổi.
  - Đặt tên hằng cũng giống như quy tắc đặt tên biến.
  - Chỉ có các kiểu dữ liệu boolean, integer, float, string mới có thể chứa các hằng.
- Cú pháp: define("TÊN_HẰNG", giá_trị);
  - Ví dụ: khai báo hằng số PI có giá trị là 3.14

```
<?php
        define("PI", 3.14);
        $r=10;
        $s= PI * pow($r ,2); → $s = 314
        $p = 2 * PI * $r; → $p = 62.8
?>
```

Kiểu dữ liêu trong PHP

- PHP hỗ trợ 4 kiểu dữ liệu
  - Kiểu số.
  - Kiểu chuỗi
  - Kiểu lôgic
  - Kiểu mảng và kiểu đối tượng

Kiểu dữ liệu – mô tả

- Boolean: chỉ có một trong hai giá trị là TRUE và FALSE
- Integer: Kiểu số nguyên. Giá trị có thể là số trong hệ thập phân, thập lục phân và bát phân.
- Float / double: Kiểu dữ liệu số thực.
- String: Kiểu dữ liệu chuỗi, ký tự. Trong đó, mỗi ký tự chiếm 1 byte.
  - Mỗi chuỗi có thể chứa một hay nhiều ký tự thuộc 256 ký tự khác nhau.
  - Chuỗi không có giới hạn về kích thước.
  - Giới hạn bởi nháy đơn (‘) hoặc kép (“)
  - Chuỗi đặt trong nháy kép bị thay thế và xử lý ký tự thoát. Trong nháy đơn thì không.
    - PHP **không** hỗ trợ Unicode, để làm việc với Unicode bạn phải sử dụng UTF8 với các hàm **utf8_encode**() – **utf8_decode**().

Kiểu dữ liệu – mô tả

- Array: Kiểu dữ liệu mảng các phần tử
- Object: Kiểu dữ liệu là đối tượng của một lớp

Chuyển đổi kiểu dữ liệu

- Trong quá trình tính toán, kiểu dữ liệu cũ của biến có thể không còn phù hợp nữa (kết quả tính toán vượt khỏi phạm vi của kiểu dữ liệu cũ) => chuyển đổi kiểu dữ liệu
- Cách thực hiện: ghi tên kiểu dữ liệu mà biến muốn chuyển đổi vào phía trước biến
  - Ví dụ:

```
<?php
        $don_gia = 5000;
        $so_luong = 100;
        $thanh_tien = (double)($so_luong*$don_gia);
?>
```

Chuyển kiểu dữ liệu

- Cú pháp
  - Cách 1 **(automatic)**

    $var = "100" + 15;

    $var = "100" + 15.0;

    $var = 39 . " Steps";
  - Cách 2: **(datatype) $var**
  - Cách 3: **settype($var, "datatype")**

$var	(int)$var	(bool)$var	(string)$var
Null	0	false	""
true	1		"1"
false	0		""
"6 feet"	6	true	
"foo"	0	true	

Các toán tử

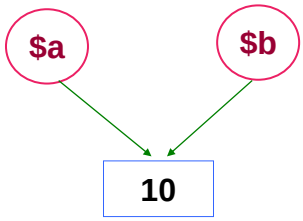
Các phép toán cơ bản:

- Phép gán: $biến = biểu_thức;
- Các phép toán số học: +, -, *, /, %
- Các phép toán so sánh: ==, ===, !=, >=, <=, >, <
- Các phép toán kết hợp: ++, --, +=, -=, *=, /=
- Các phép toán logic: !, &&, ||
- Toán tử tam phân: (điều_kiện **?** giá_trị_1 **:** giá_trị_2)
- Phép ghép chuỗi: **.** (dấu chấm)
- Toán tử error: @, ngăn không cho thông báo lỗi
  - Ví dụ: $a = 10; $b = 0; $c = @$a / $b;

Tham chiếu trong PHP

- Trong PHP tham chiếu có nghĩa là lấy cùng một giá trị bằng nhiều tên biến khác nhau.
- Ký hiệu tham chiếu: **&**

```
<?php
        $a = 10;
        $b = &$a;
        echo $a; → 10
        echo $b; → 10
?>
```



Cấu trúc điều khiển trong PHP

- Cấu trúc rẽ nhánh
- Cấu trúc chọn lựa
- Cấu trúc lặp
- Sử dụng break và continue trong cấu trúc lặp

Cấu trúc rẽ nhánh if

```
if (điều kiện)
{
        khối lệnh
}
```

```
if(điều kiện)
    {
            khối lệnh 1
    }
else
    {
            khối lệnh 2
    }
```

– Điều kiện là một biểu thức logic trả về đúng (TRUE) hoặc sai (FALSE)

Toán tử ?:

- Cú pháp:

  (điều kiện)?<kết quả khi điều kiện đúng>:<kết quả khi điều kiện sai>

  – Ý nghĩa: dùng để thay thế cho cấu trúc điều khiển if…else với một câu lệnh bên trong

  – Có thể lồng nhiều toán tử ?: với nhau

Cấu trúc rẽ nhánh if

- **Dạng: if … elseif … else**
  - Cú pháp:

```
if(điều kiện 1)
        {
                khối lệnh 1
        }
elseif(điều kiện 2)
        {
                khối lệnh 2
        }
…
else
        {
                khối lệnh khi không thỏa các điều kiện trên
        }
```


Cấu trúc chọn lựa switch

- Cú pháp

```
switch(biến điều kiện)
{
        case giá trị 1:
                khối lệnh 1
                break;
        case giá trị 2:
                khối lệnh 2
                break;
        ...
        [default: khối lệnh khi không thỏa tất cả các case trên]
}
```

Cấu trúc lặp

- Cấu trúc lặp cho phép thực hiện nhiều lần một khối lệnh của chương trình khi thỏa điều kiện
- Gồm có các cấu trúc: for, foreach, while, do…while

Cấu trúc lặp for

- Công dụng:
  - for thường sử dụng khi chúng ta biết trước số lần cần lặp, biến đếm chạy trong khoảng giới hạn của vòng lặp, và giá trị lặp.
- Cú pháp:

```
for($biến_đếm = giá trị khởi đầu của vòng lặp for; điều kiện giới hạn của vòng lặp for; giá trị lặp của vòng lặp for)
{
        khối lệnh
}
```

Cấu trúc lặp foreach

- Công dụng:
  - foreach duyệt từ phần tử đầu tiên đến phần tử cuối cùng của tập hợp (mảng) để thực hiện các xử lý tương ứng
- Cú pháp duyệt giá trị các phần tử trong mảng:

```
foreach ($ten_mang as $gia_tri)
{
        khối lệnh
}
```

- Cú pháp duyệt cả khóa và giá trị các phần tử trong mảng:

```
foreach ($ten_mang as $tu_khoa => $gia_tri)
{
        khối lệnh
}
```


Cấu trúc lặp while và do … while

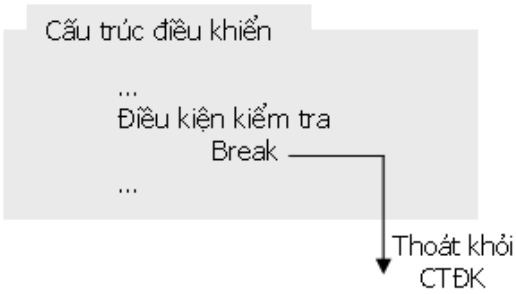
- Công dụng
  - Thực hiện lặp đi lặp lại một công việc nào đó khi thỏa điều kiện.
  - Được sử dụng khi không xác định được số lần lặp (số lần lặp phụ thuộc vào điều kiện tại thời điểm thực thi)
- Cú pháp

```
while(điều kiện)
{
        khối lệnh
}
```

```
do
{
    khối lệnh
}
while(điều kiện);
```

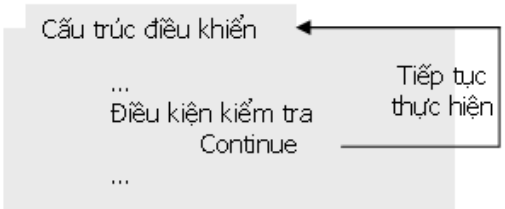
Sử dụng break và continue trong cấu trúc lặp

- break
  - Công dụng: thoát khỏi cấu trúc điều khiển dựa trên kết quả của biểu thức luận lý kèm theo (điều kiện kiểm tra)



Sử dụng break và continue trong cấu trúc lặp

- continue
  - Công dụng: Khi gặp continue, các lệnh bên dưới continue tạm thời không thực hiện tiếp, khi đó con trỏ sẽ nhảy về đầu vòng lặp để kiểm tra giá trị của biểu thức điều kiện còn đúng hay không.
  - continue thường đi kèm với một biểu thức luận lý.



Hàm

- Hàm thư viện
- Hàm do người dùng tự định nghĩa

Hàm thư viện

- Là các hàm được PHP xây dựng sẵn, chỉ cần gọi khi sử dụng
- Có rất nhiều nhóm hàm trong thư viện: nhóm hàm chuỗi, toán học, thời gian, lịch, mảng, thư mục, tập tin, mail, CSDL, …
- Cách dùng: viết đúng tên hàm và truyền vào các giá trị cần thiết.
- Ví dụ: dùng hàm round() để làm tròn một số, hàm date() để lấy ngày hiện tại có định dạng ngày tháng năm

Hàm do người dùng tự định nghĩa

- Cú pháp:

```
function Tên_hàm(Danh_sách_các_tham_số)
{
        khối lệnh bên trong hàm
        return giá_trị;
}
```

- Trong đó
  - Tên hàm: tên hàm nên có ý nghĩa gợi nhớ
  - Danh sách các tham số: không cần xác đinh kiểu, hàm có thể có hoặc không có tham số
  - Giá trị: Hàm có thể có hoặc không có giá trị trả về

Hàm do người dùng tự định nghĩa

- Phân loại tham số của hàm: có hai loại
  - Tham trị: truyền tham số theo giá trị
    - Tham số truyền giá trị từ bên ngoài vào cho hàm.
    - Khi thay đổi giá trị của tham trị bên trong hàm thì giá trị của nó ở ngoài hàm vẫn không bị thay đổi.
  - Tham biến: truyền tham số theo địa chỉ
    - Tham số truyền giá trị từ bên ngoài cho hàm và trả giá trị ở trong hàm ra bên ngoài.
    - Khi thay đổi giá trị của tham biến bên trong hàm thì giá trị của nó ở ngoài hàm cũng sẽ thay đổi sau khi chúng ta gọi hàm đã xây dựng.
    - Đối với tham biến chúng ta sẽ sử dụng cú pháp với ký tự & ở phía trước.

Hàm do người dùng tự định nghĩa

- Tham số tùy chọn
  - Là những tham số có thể được truyền giá trị hoặc không
  - Cho phép tạo sẵn các giá trị mặc định cho tham số
  - Những tham số này chỉ xuất hiện cuối danh sách các tham số
  - Cú pháp:

```
function Tên_hàm(Danh sách các tham trị, tham biến,
  $tham_số_tùy_chọn=giá_trị)
    {
        khối lệnh bên trong hàm
        return giá_trị;
    }
```


Các hàm kiểm tra giá trị của biến

- Kiểm tra tồn tại isset()
- Kiểm tra giá trị rỗng empty()
- Kiểm tra trị kiểu số is_numeric()
- Kiểm tra kiểu dữ liệu của biến
- Xác định kiểu của biến gettype()

Kiểm tra tồn tại: isset()

- Ý nghĩa: dùng để kiểm tra biến có giá trị hay không
- Cú pháp: isset(<tên biến 1>, <tên biến 2>, …)
- Kết quả trả về:
  - TRUE: nếu tất cả các biến đều có giá trị
  - FALSE: nếu một biến bất kỳ không có giá trị

```
<?php
        if(isset($_POST[”ten_dang_nhap”]))
                echo ”Xin chào ”.$_POST[”ten_dang_nhap”];
        else
                echo ”Vui lòng nhập tên đăng nhập”;
?>
```


Kiểm tra giá trị rỗng: empty()

- Ý nghĩa: dùng để kiểm tra biến có giá trị rỗng hay không
- Cú pháp: empty(<tên biến>)
  - Kết quả trả về:
    - TRUE: nếu biến có giá trị rỗng
    - FALSE: nếu một biến có giá trị khác rỗng
  - Các giá trị được xem là rỗng:
    - “” (chuỗi rỗng), NULL
    - 0 (khi kiểu là integer), FALSE, array()
    - var $var (biến trong lớp được khai báo nhưng không có giá trị)

Kiểm tra trị kiểu số: is_numeric()

- Ý nghĩa: dùng để kiểm tra biến có giá trị kiểu số hay không
- Cú pháp: is_numeric(<tên biến>)
  - Kết quả trả về:
    - TRUE: nếu biến có giá trị kiểu số
    - FALSE: nếu biến có giá trị không phải kiểu số

```
<?php
        if(is_numeric($_POST[“so_luong”]))
        {
                $so_luong = $_POST[“so_luong”];
                $thanh_tien = $so_luong * $don_gia;
        }
        else
                echo “Số lượng phải là kiểu số!”;
?>
```


Kiểm tra kiểu dữ liệu của biến

- is_int() / is_long()
- is_string()
- is_double()
- is_array()
- is_bool()
- is_file()
- is_float()
- is_null()
- is_object()
  - Ý nghĩa: kiểm tra giá trị của biến có phải là kiểu integer - long – string – double hay không
  - Cú pháp chung: tên_hàm(<tên_biến>)

Xác định kiểu của biến: gettype()

- Ý nghĩa; kiểm tra biến hoặc giá trị có kiểu dữ liệu nào: integer, string, double, array, object, class, …
- Cú pháp: gettype(<tên biến> hoặc <giá trị>)
- Kết quả trả về: kiểu của giá trị hay kiểu của biến

```
<?php
        $n = "Đây là chuỗi";
        $a = 123;
        $b = 123.456;
        $mang = array(1,2,3);
        echo gettype($n); → string
        echo gettype($a); → integer
        echo gettype($b); → double
        echo gettype($mang); → array
?>
```


Các hàm toán học

- Giá trị tuyệt đối
- Pi
- Lũy thừa
- Phát sinh ngẫu nhiên
- Làm tròn
- Tính căn

Các hàm toán học

- abs(): kết quả trả về là giá trị tuyệt đối của một số
  - Cú pháp: abs(số)

```
<?php
echo abs(-4.2); → 4.2 (kiểu float/double)
echo abs(-5); → 5 (kiểu integer)
echo abs(4); → 4 (kiểu integer)
?>
```

- pi(): kết quả trả về là hằng số PI trong toán học
  - Cú pháp: pi()

```
<?php
echo pi(); → 3.1415926535898
?>
```


Các hàm toán học

- pow(): kết quả trả về là phép tính lũy thừa của một số theo một số mũ chỉ định
  - Cú pháp: pow(số, lũy_thừa)

```
<?php
echo pow(2,5); → 32
?>
```

- rand(): kết quả trả về là một giá trị số nguyên ngẫu nhiên bất kỳ (nằm trong khoảng từ 0 đến 32768)
  - Khi hàm rand() có hai tham số là giá trị nhỏ nhất (min) và giá trị lớn nhất (max) thì giá trị trả về sẽ là một giá trị số nằm trong khoảng từ giá trị nhỏ nhất đến giá trị lớn nhất.
  - Cú pháp: rand ([min, max])

```
<?php
echo rand(); → 10325
echo rand(); → 7774
echo rand(10, 100); → 57
?>
```

Các hàm toán học

- round(): kết quả trả về là một số đã được làm tròn.
  - Nếu chỉ có một tham số là số được truyền vào thì kết quả trả về là một số nguyên.
  - Nếu có cả hai tham số là số và vị trí làm tròn thì kết quả trả về là một số được làm tròn dựa trên vị trí làm tròn.
- Cú pháp: round (số [, vị trí làm tròn])
  - Vị trí làm tròn là một số nguyên âm hoặc nguyên dương dùng để chỉ định vị trí muốn làm tròn, được tính từ vị trí dấu chấm thập phân

```
<?php
echo round(3.4); → 3
echo round(3.5); → 4
echo round(577.545, 2); → 577.55
echo round(577.545,-1); → 580
?>
```


Các hàm toán học

- sqrt(): tính căn bậc hai của một số dương bất kỳ
  - Kết quả trả về là căn bậc hai của số
  - Cú pháp: sqrt(số)

```
<?php
echo sqrt(9); → 3
?>
```

Các hàm chuỗi

- Định dạng chuỗi
- So sánh chuỗi
- Tìm kiếm và thay thế
- Kết hợp và tách chuỗi

Định dạng chuỗi - Bỏ ký tự thừa

- ltrim()
  - Loại bỏ các ký tự thừa bên trái chuỗi.
  - Kết quả trả về là một chuỗi không có ký tự thừa bên trái chuỗi.
  - Cú pháp:

    ltrim(str [, charlist])
    - Tham số bắt buộc là chuỗi str
    - Tham số tùy chọn là danh sách các ký tự muốn loại bỏ bên trái chuỗi.

```
<?php
        $text = "\t\tXin chào các bạn :)      ...";
        echo ltrim($text);
        → "Xin chào các bạn :) ..."
?>
```

Định dạng chuỗi - Bỏ ký tự thừa

- rtrim(), chop()
  - Loại bỏ các ký tự thừa bên phải chuỗi.
  - Kết quả trả về là một chuỗi không có ký tự thừa bên phải chuỗi.
  - Cú pháp:

    rtrim(str [, charlist])

    chop(str [, charlist])

```
<?php
        $text = "\t\tXin chào các bạn :)      ...  ";
        echo rtrim($text);
        → "\t\tXin chào các bạn :) ..."
        echo chop($text, "\t.");
        → "Xin chào các bạn :)"
?>
```


Định dạng chuỗi - Bỏ ký tự thừa

- trim()
  - Loại bỏ các ký tự thừa bên phải và bên trái chuỗi.
  - Kết quả trả về là một chuỗi không có ký tự thừa bên phải và bên trái chuỗi.
  - Cú pháp:

    trim(str [, charlist])

```
<?php
        $text = "\t\tXin chào các bạn :)      ...  ";
        echo trim($text);
        → "Xin chào các bạn :) ..."
        echo trim($text, "\t.");
        → "Xin chào các bạn :)"
?>
```

Định dạng chuỗi - Bỏ ký tự thừa

- Lưu ý: nếu không có tham số thứ hai thì hàm ltrim(), rtrim(), chop() sẽ tự động loại bỏ các ký tự sau:
  - " " (ASCII 32 (0x20)), ký tự khoảng trắng.
  - "\t" (ASCII 9 (0x09)), ký tự tab.
  - "\n" (ASCII 10 (0x0A)), ký tự về đầu dòng mới.
  - "\r" (ASCII 13 (0x0D)), ký tự xuống dòng.
  - "\0" (ASCII 0 (0x00)), byte NULL.
  - "\x0B" (ASCII 11 (0x0B)), tab dọc.
- Ngoài ra, có thể liệt kê các ký tự muốn loại bỏ ở charlist

Định dạng chuỗi – Chuyển đổi kiểu chữ

- strtoupper(): chuyển đổi chữ thành chữ HOA
- strtolower(): chuyển đổi chữ thành chữ thường
- ucfirst(): viết hoa ký tự đầu tiên của chuỗi
- ucwords(): viết hoa ký tự đầu tiên của mỗi từ
- Cú pháp

  strtoupper(str)

  strtolower(str)

  ucfirst(str)

  ucwords(str)

```
<?php
        echo strtoupper("happy new year");
        → "HAPPY NEW YEAR"
        echo strtolower("HAPPY NEW YEAR");
        → "happy new year"
        echo ucfirst("happy new year");
        → "Happy new year"
        echo ucwords("happy new year");
        → "Happy New Year"
?>
```

Định dạng chuỗi – Chiều dài chuỗi

- strlen()
  - Kết quả trả về là chiều dài của một chuỗi
  - Cú pháp

    strlen(str)
  - Ví dụ:

```
<?php
        echo strlen(“abcdef”); → 6
?>
```


Định dạng chuỗi – Đếm ký tự

- **count_chars()**
  - Kết quả trả về là một mảng với các phần tử có khóa là mã ASCII và giá trị là số lần xuất hiện của các ký tự trong chuỗi.
  - Thứ tự của các khóa được sắp tăng dần theo bảng mã ASCII
  - Cú pháp

    count_chars(str [, mode])

Định dạng chuỗi – Đếm ký tự

- Mode: là kiểu đếm (tham số tùy chọn)
- Mode = 0 (hoặc không sử dụng mode): liệt kê tất cả các ký tự ASCII từ 0 đến 255 và số lần xuất hiện.
- Mode = 1 : chỉ liệt kê danh sách có số lần xuất hiện lớn hơn 0
- Mode = 2: chỉ liệt kê danh sách có số lần xuất hiện bằng 0
- Mode = 3: chuỗi mới mà mỗi ký tự trong chuỗi gốc chỉ xuất hiện ở chuỗi mới 1 lần theo thứ tự tăng dần của bảng mã ASCII.
- Mode = 4: trả về một chuỗi mới với các ký tự theo thứ tự tăng dần theo bảng mã ASCII, mà các ký tự này không xuất hiện ở chuỗi gốc.

Định dạng chuỗi – Đếm ký tự

- Ví dụ: mode = 1

```
<?php
        $str = "Hello World!";
        print_r(count_chars($str,1));
        →
        Array
        (
        [32] => 1 [33] => 1 [72] => 1
        [87] => 1 [100] => 1 [101] => 1
        [108] => 3 [111] => 2 [114] => 1
        )
?>
```

So sánh chuỗi – Có phân biệt HOA - thường

- strcmp()
  - So sánh hai chuỗi có phân biệt chữ HOA, chữ thường.
  - Kết quả trả về:
    - =0: nếu hai chuỗi bằng nhau
    - <0: nếu chuỗi str1 nhỏ hơn chuỗi str2
    - >0: nếu chuỗi str1 lớn hơn chuỗi str2
- Cú pháp

  strcmp(str1, str2)
- Lưu ý: Việc so sánh chuỗi dựa trên bảng mã ASCII của các ký tự

So sánh chuỗi – Không phân biệt HOA - thường

- strcasecmp(), strnatcmp()
  - So sánh hai chuỗi không phân biệt chữ HOA, chữ thường.
  - Kết quả trả về:
    - =0: nếu hai chuỗi bằng nhau
    - <0: nếu chuỗi str1 nhỏ hơn chuỗi str2
    - >0: nếu chuỗi str1 lớn hơn chuỗi str2
- Cú pháp

  strcasecmp(str1, str2)

  strnatcmp(str1, str2)

```
<?php
        echo strnatcmp("hello", "Hello"); → 0
?>
```

Tìm kiếm và thay thế - Tìm một chuỗi trong chuỗi

- strstr(), strchr()
  - Tìm một chuỗi trong chuỗi.
  - Kết quả trả về là một chuỗi con của chuỗi str1 được lấy từ vị trí xuất hiện đầu tiên của chuỗi str2 đến hết chuỗi str1 nếu tìm thấy chuỗi str2 trong chuỗi str1, nếu không tìm thấy chuỗi str2 trong chuỗi str1 thì trả về FALSE.
- Cú pháp

  strstr(str1, str2)

  strchr(str1, str2)

```
<?php
        $email = “phuong@yahoo.com”;
        $domain = strstr($email, “@”);
        echo $domain; →“@yahoo.com”
?>
```


Tìm kiếm và thay thế - Tìm vị trí của chuỗi con

- strpos()
  - Kết quả trả về là vị trí tìm thấy của chuỗi str2 trong chuỗi str1, nếu trong chuỗi str1 có nhiều chuỗi str2 thì kết quả trả về sẽ là vị trí đầu tiên mà chuỗi str2 xuất hiện trong chuỗi str1. Nếu không tìm thấy chuỗi str2 trong chuỗi str1 thì kết quả trả về là FALSE
- Cú pháp

  strpos(str1, str2)

- Lưu ý: Để kiểm tra kết quả thì phải dùng toán tử === thay cho toán tử ==

Tìm kiếm và thay thế

- str_replace()
  - Tìm kiếm chuỗi str1 trong chuỗi str, nếu tìm thấy thì thay thế chuỗi str1 bằng chuỗi str2 trong chuỗi str.
- Cú pháp

  str_replace(str1, str2, str)

```
<?php
        echo str_replace("em", "bé", "Hôm qua em đến trường");
        → "Hôm qua bé đến trường"
?>
```


Kết hợp và tách chuỗi – Tách chuỗi

- explode()
  - Tách chuỗi str thành nhiều chuỗi con bằng cách chỉ định chuỗi tách separator và gán từng chuỗi con này vào các phần tử của mảng
- Cú pháp

  explode(separator, str)

```
<?php
        $chuoi = "happy new year";
        $mang_chuoi = explode(" ", $chuoi);
        echo $mang_chuoi[0]; → "happy"
        echo $mang_chuoi[1]; → "new"
        echo $mang_chuoi[2]; → "year"
?>
```

Kết hợp và tách chuỗi – Kết hợp chuỗi

- implode()
  - Kết hợp các phần tử của mảng thành một chuỗi và các phần tử khi ráp thành chuỗi sẽ cách nhau bằng chỉ định cách separator
- Cú pháp

  implode(separator, array)

```
<?php
        $mang_chuoi = array("happy", "new", "year");
        $chuoi = implode(" ", $mang_chuoi);
        echo $chuoi; → "happy new year"
?>
```


Các hàm thời gian

- Kiểm tra ngày hợp lệ
- Định dạng ngày
- Lấy các giá trị của ngày hiện tại
- Lấy thời gian hiện tại
- Chuyển đổi ngày
- Chuyển chuỗi thời gian thành giá trị thời gian
- Đổi thời gian sang đơn vị giây
- Định dạng thời gian thành số nguyên

Kiểm tra ngày hợp lệ

- checkdate()
  - Kiểm tra ngày với tháng, ngày, năm được truyền vào, nếu ngày hợp lệ thì kết quả trả về là TRUE, ngược lại trả về FALSE
- Cú pháp:

  checkdate(tháng, ngày, năm)

```
<?php
        echo checkdate(11,30,2007); → TRUE
?>
```


Định dạng ngày

- date()
  - Kết quả trả về là ngày hiện tại sau khi đã được định dạng
- Cú pháp:

  date(chuỗi định dạng)

```
<?php
        echo date("d/m/Y h:m:s");
        → 28-11-2007 11:11:07
        echo date("D-M-Y");
        → Thu–Dec-2007
?>
```

Lấy các giá trị của ngày hiện tại

- getdate()
  - Kết quả trả về là một mảng gồm 10 phần tử (có khóa dạng chuỗi) chứa các thông tin cần thiết của ngày hiện tại.

  [seconds] – giây, [minutes] - phút, [hours] – giờ

  [mday] – ngày trong tháng, [wday] – ngày trong tuần

  [mon] – tháng (dạng số), [year] – năm, [yday] – ngày trong năm

  [weekday] – thứ trong tuần, [month] – tháng (dạng chữ)

Lấy các giá trị của ngày hiện tại

- Cú pháp:

  getdate()

- Ví dụ:

```
<?php
        $today = getdate();
        print_r($today);
?>
        → Array
        (       [seconds] => 18 [minutes] => 18 [hours] => 13
                [mday] => 28 [wday] => 3 [mon] => 11 [year] => 2007
                [yday] => 331 [weekday] => Wednesday
                [month] => November
        )
```

Lấy thời gian hiện tại

- time()
  - Kết quả trả về là thời gian hiện tại được đo lường theo giá trị giây (tính từ 1/1/1970 00:00:00 GMT)
- Cú pháp:

time()

```
<?php
        $tuan_sau = time() + (7 * 24 * 60 * 60);
        // 7 days; 24 hours; 60 mins; 60secs
        echo ”Hiện tại:”. date(”Y-m-d”) ."<br>";
        → Hiện tại: 2007-11-28
        echo ”Tuần sau: ”. date(”Y-m-d”, $tuan_sau) ."<br>";
        → Tuần sau: 2007-12-05
?>
```


Chuyển chuỗi thời gian thành giá trị thời gian

- strtotime()
  - Kết quả trả về là tổng số giây của thời gian nếu chuỗi thời gian đúng, ngược lại sẽ trả về FALSE
  - Tổng số giây = thời gian được truyền vào trong hàm strtotime() - January 1 1970 00:00:00 (vì thời gian bắt đầu được tính từ: January 1 1970 00:00:00)
- Cú pháp:

strtotime(chuỗi thời gian)

```
<?php
        echo strtotime("now");
        → 1.204.795.981
        // Với now = 06-03-2008 16:33:01
?>
```