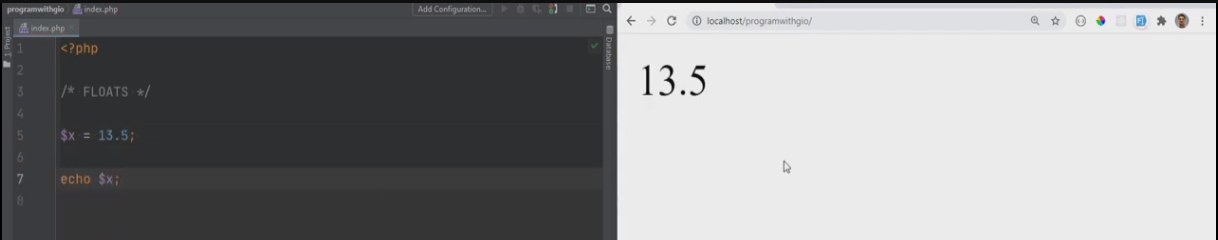
1.7 Kiểu dữ liệu Float trong PHP - Hướng dẫn đầy đủ về PHP 8

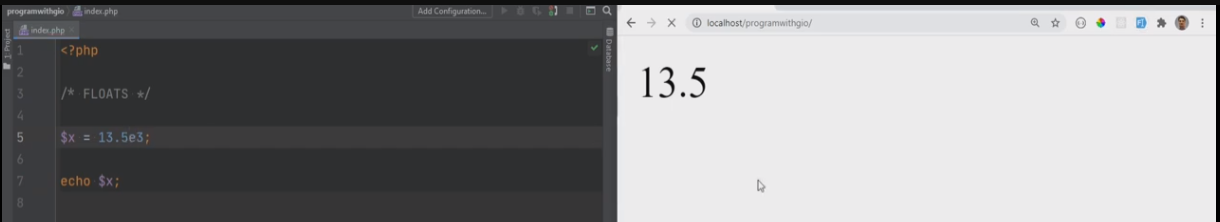
* Số dấu chấm động và phần thập phân

Các số dấu chấm động là những số có phần thập phân, vì vậy chúng ta có thể cho x bằng 13.5 và nếu chúng ta in ra x, chúng ta sẽ nhìn thấy 13.5.



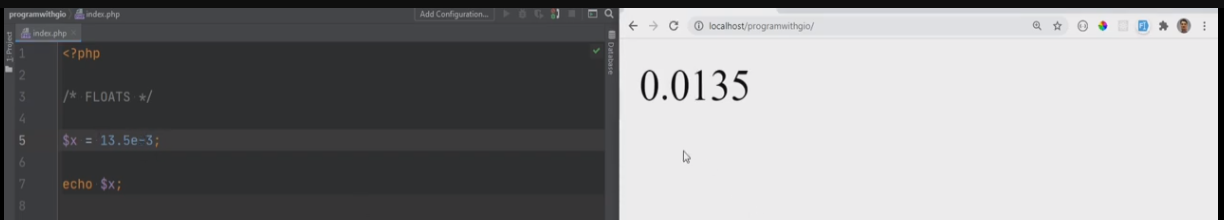
* Biểu diễn số dấu chấm động dưới dạng mũ

Bạn cũng có thể biểu diễn số dấu chấm động dưới dạng mũ, ví dụ như chúng ta có thể biểu diễn 13.5E3, và nó sẽ tương đương với 13,500.



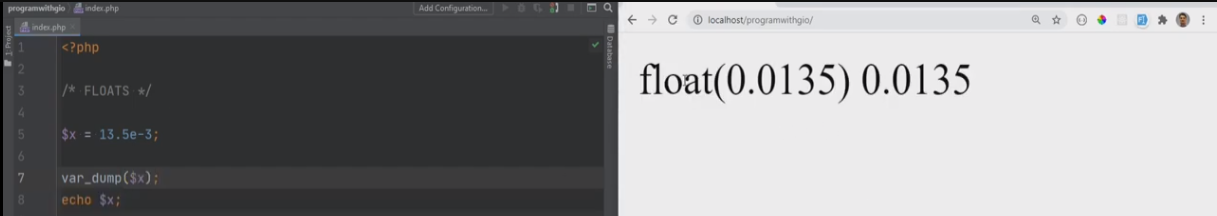
* Biểu diễn số dấu chấm động âm

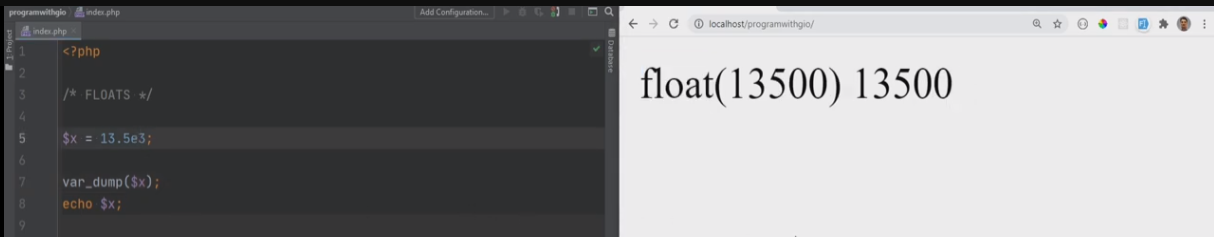
Chúng ta cũng có thể làm với số âm, ví dụ như E3, và điều này sẽ cho kết quả là 0,0135.



* Kiểm tra kiểu dữ liệu float của biến x bằng hàm var\_dump

Kiểu dữ liệu của x cũng là float, và chúng ta có thể kiểm tra điều đó bằng cách sử dụng hàm var\_dump và xác nhận rằng đó là một số dấu chấm động.

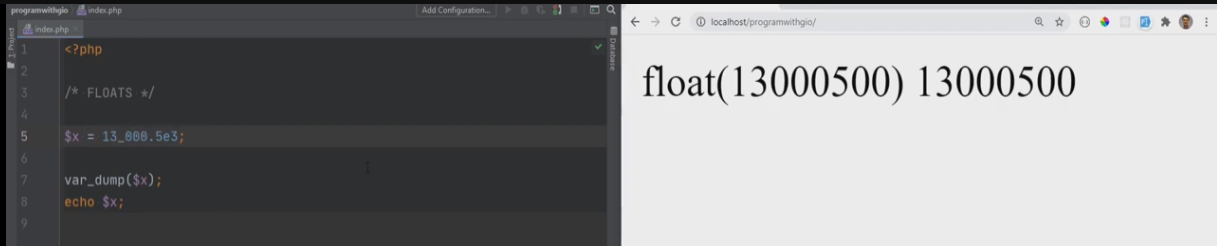
Ngay cả khi chúng ta sử dụng số dương mà không chứa phần thập phân, nó vẫn là số dấu chấm động.

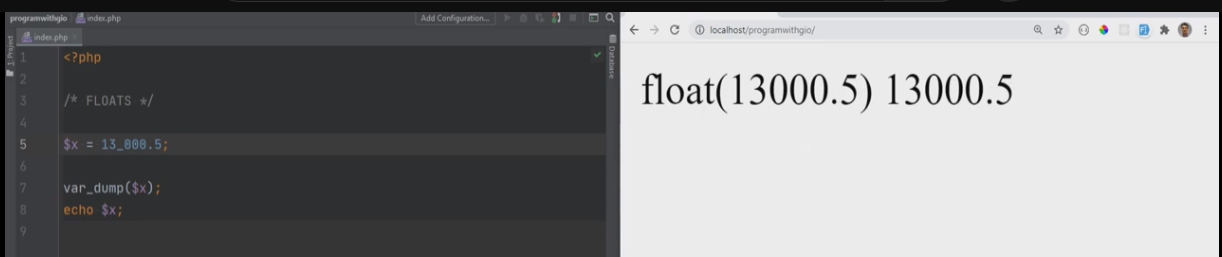


* Sử dụng dấu gạch dưới trong số dấu chấm động để tăng tính đọc hiểu

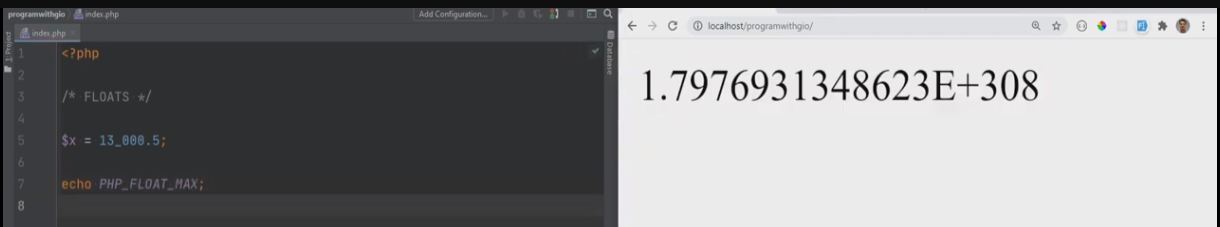
Ngoài ra, như đã thảo luận với số nguyên, kể từ PHP 7.4, bạn cũng có thể sử dụng dấu gạch dưới trong các số của mình để tăng tính đọc hiểu.

Vậy bạn có thể có một số như 13,000, và trong đó bạn có thể thêm dấu gạch dưới, và điều này hoàn toàn hợp lệ.

 Và nếu chúng ta làm tươi trang, chúng ta sẽ nhận được kiểu dữ liệu float và chúng ta có thể loại bỏ dạng mũ trong đó và làm tươi trang, và đó là cách nó hoạt động.

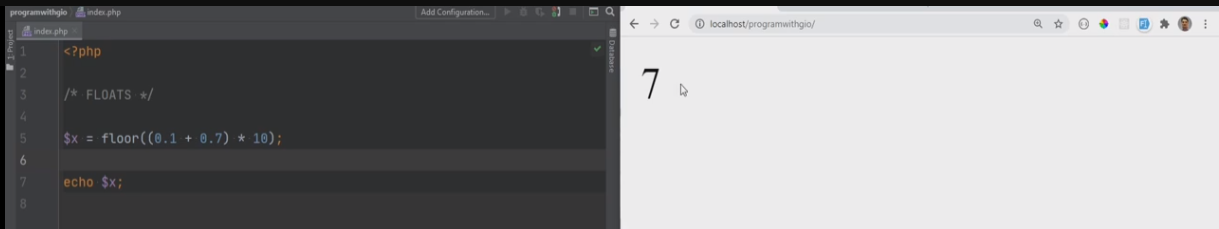
 Tương tự như số nguyên, kích thước của số dấu chấm động phụ thuộc vào nền tảng, và bạn có thể kiểm tra điều đó bằng cách sử dụng các hằng số được định nghĩa trước đó.

* Hằng số PHP\_FLOAT\_MAX và PHP\_FLOAT\_MIN

Các hằng số đó là PHP\_FLOAT\_MAX và PHP\_FLOAT\_MIN, tôi tin là như vậy. Vậy hãy kiểm tra PHP\_FLOAT\_MAX và PHP\_FLOAT\_MIN, để chúng ta có thể sử dụng chúng, hãy in ra giá trị tối đa. 

* Giới hạn độ chính xác của số dấu chấm động

Và chúng ta sẽ thấy đó là một số rất lớn. Điều quan trọng cần nhớ khi làm việc với các số dấu chấm động là độ chính xác giới hạn của chúng.

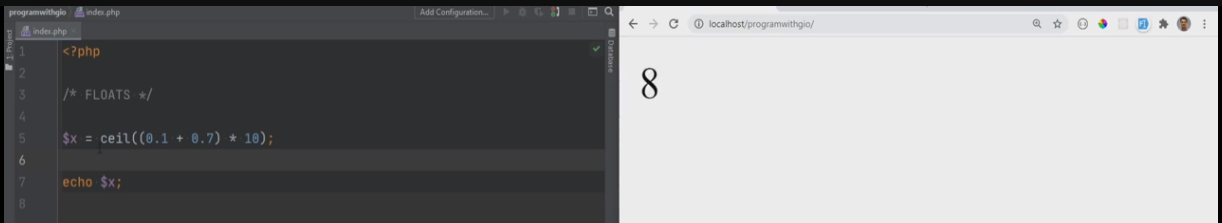
Ví dụ, nếu chúng ta lấy biểu thức này, floor((0.1 + 0.7) \* 10), bạn có nghĩ rằng nó sẽ trả về 8, đúng không? Bởi vì 0.1 + 0.7 là 0.8, nhân với 10 thì được 8, và floor sẽ làm tròn xuống thành 8 và trả về giá trị 8. Nếu chúng ta load lại trang, chúng ta sẽ nhận được 7.  Và lý do cho điều đó là vì 0.1 hoặc 0.7 không có một biểu diễn chính xác trong hệ cơ số 2, còn được gọi là hệ nhị phân, và hệ nhị phân được sử dụng bên trong để lưu trữ các số dấu chấm động. Vì vậy, khi chuyển đổi nội bộ thành hệ nhị phân này, nó mất một số độ chính xác. Trong trường hợp này, 0.1 + 0.7 \* 10 thực sự bằng số này ở đây.

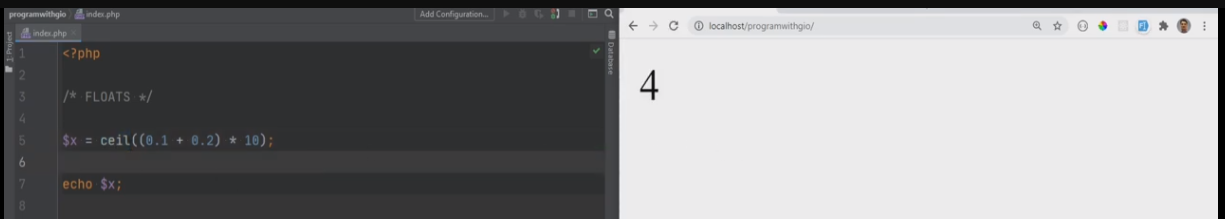


Đúng, như bạn biết, hàm floor đơn giản là làm tròn xuống cho tất cả các số. Trong trường hợp này, khi áp dụng floor cho số này, nó chỉ loại bỏ toàn bộ phần thập phân và chúng ta chỉ còn lại 7. Vì vậy, tại đây tôi in ra số 7. Bây giờ, tôi biết rằng tôi đã giới thiệu nhiều khái niệm ở đây, chẳng hạn như phép nhân, phép cộng, phương thức floor, ….

* Hàm ceiling và vấn đè của việc làm tròn lên

Đừng lo lắng quá nhiều về điều này vào lúc này. Chúng ta sẽ đi qua các toán tử trong một video riêng. Tôi chỉ muốn thông báo cho bạn những điều cần lưu ý khi làm việc với số dấu chấm động. Như bạn đã biết, hàm floor làm tròn xuống cho mọi số. Và ngược lại, hàm ceiling hoặc ceil làm tròn lên cho mọi số. Nếu chúng ta làm tươi trang trang này, chúng ta sẽ nhận được số tám, điều đó là đúng như mong đợi.

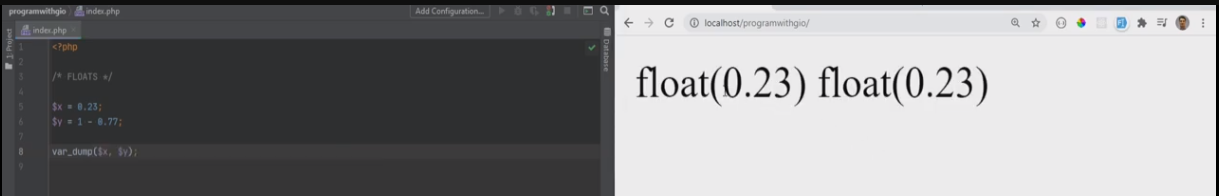
Bây giờ chúng ta hãy xem xét một ví dụ khác với hàm ceiling mà kết quả sẽ không như mong đợi. Vì vậy, hãy thay đổi biểu thức thành (0.1 + 0.2) \* 10. Và bạn có nghĩ rằng điều này sẽ trả về số ba vì 0.1 + 0.2 là 0.3, nhân với 10 là ba. Và sau đó, ceiling sẽ trả về số ba, nhưng trong thực tế, kết quả là 4.

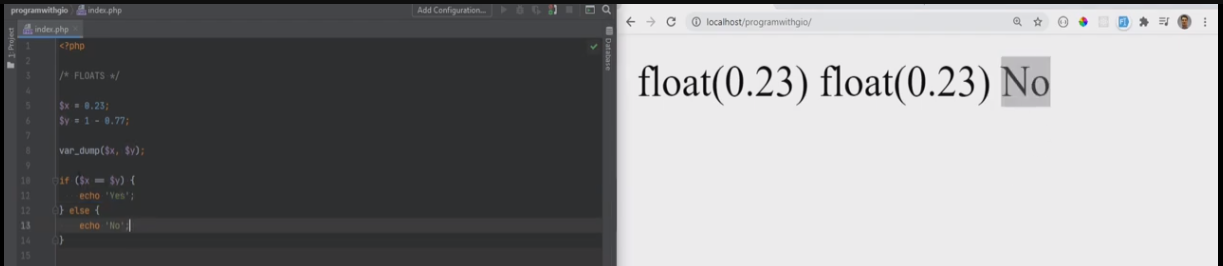


Nếu giá trị này là không, thì nó sẽ không được làm tròn lên thành bốn, nó sẽ vẫn là ba.

* Cảnh báo về tính chính xác và so sánh số dấu chấm động.

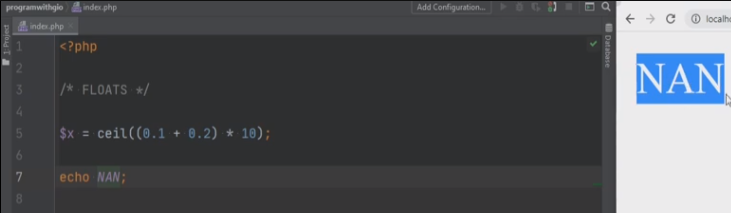
Vì vậy, cơ bản là không bao giờ tin tưởng vào số dấu chấm động cho đến chữ số cuối cùng và không bao giờ so sánh trực tiếp các số dấu chấm động để kiểm tra tính bằng nhau. Tôi sẽ để lại một liên kết trong phần mô tả mà bạn có thể đọc thêm về số dấu chấm động và cách so sánh chúng một cách chính xác để kiểm tra tính bằng nhau. Để tôi chỉ cho bạn ý nghĩa của việc không so sánh số dấu chấm động trực tiếp. Hãy giả sử bạn có x = 0.23 và sau đó y = 1 - 0.77. Và hãy in ra giá trị của chúng.

Chúng ta thấy rằng cả hai đều giống nhau, phải không? Cả hai đều bằng 0.23. Nhưng nếu chúng ta thực hiện so sánh đó? Vì vậy, nếu chúng ta thực hiện if x == y, và đừng lo lắng về điều kiện if trong lúc này, chúng ta sẽ nói về nó trong một video riêng. Nhưng nếu chúng ta làm như vậy, hãy làm echo "yes", ngược lại là echo "no". Bạn nghĩ rằng điều này sẽ in ra "yes", đúng không? Bởi vì về mặt kỹ thuật, chúng giống nhau. Nhưng nếu chúng ta làm tươi trang, chúng ta sẽ nhận được "no".

Đó chính là điều tôi muốn nói là bạn không nên so sánh số dấu chấm động trực tiếp như thế này.

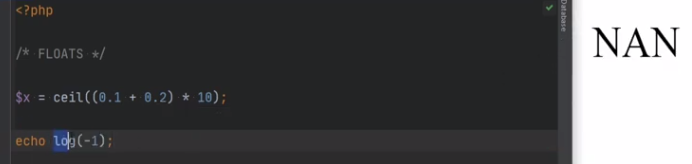
* Hằng số NaN (Not a Number)

Một điều cần lưu ý khác là một số phép toán hoặc tính toán có thể cho kết quả là giá trị không xác định, và nó sẽ được biểu diễn bằng một hằng số gọi là NaN (Not a Number).



Đúng, nếu chúng ta echo một hằng số none, chúng ta sẽ thấy giá trị là "none". Như bạn biết, "none" đại diện cho "not a number" (không phải là số), và bạn có thể nhận được giá trị này khi một số phép toán hoặc tính toán không thể được tính toán và kết quả là "not a number".

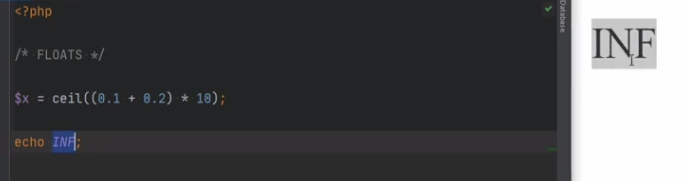
Ví dụ về một phép tính như vậy là log(-1), và điều này sẽ cho kết quả là "none".



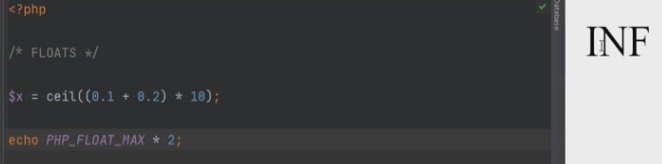
* Hằng số vô cùng (Infinity)

Đúng, còn một hằng số khác được gọi là "infinity" (vô cùng), được ký hiệu là "INF".

Nếu chúng ta echo giá trị này, chúng ta sẽ nhận được "INF".

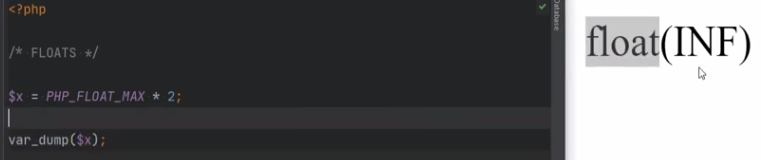
 Chúng ta sẽ nhận được giá trị này khi vượt quá giới hạn của số dấu chấm động.

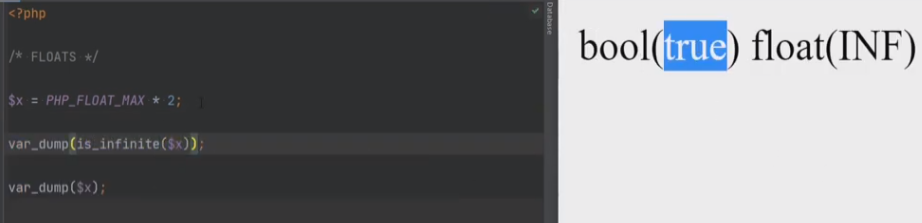
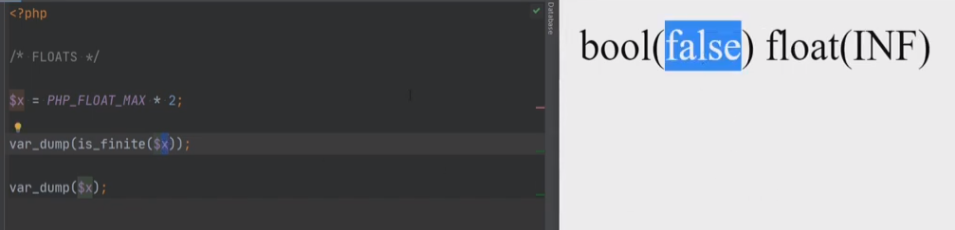
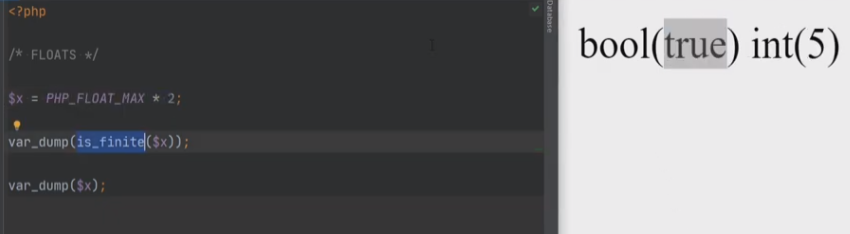
Ví dụ, nếu bạn có PHP\_FLOAT\_MAX \* 2, bạn sẽ nhận được vô cùng (INF).

 Bởi vì số đó không thể thực sự được biểu diễn, vì đây là giá trị dấu chấm động tối đa có thể lưu trữ trên nền tảng và khi nhân đó với 2, nó sẽ được in ra là vô cùng tại đây. Đúng, bất kỳ phép tính nào đẩy số dấu chấm động vượt quá giới hạn sẽ cho kết quả là vô cùng (INF).

* Cách kiểm tra NaN và Infinity trong PHP sử dụng các hàm is\_nan và is\_infinity

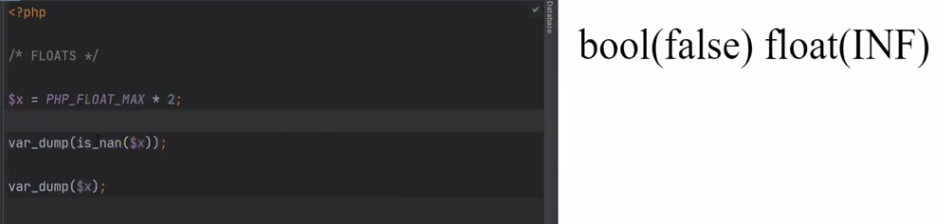
Bạn cũng không nên so sánh trực tiếp một biến với vô cùng (INF) hoặc không phải là số (NaN) để kiểm tra xem biến có phải là vô cùng hay không phải là số. Thay vào đó, bạn nên sử dụng các hàm tích hợp sẵn được gọi là is\_nan và is\_infinity.

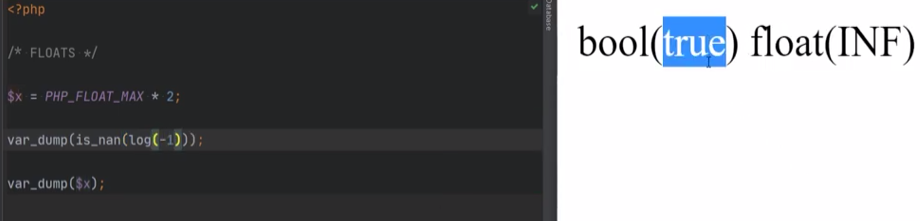
Ví dụ, nếu chúng ta gán giá trị này cho x, hãy xóa đi phần này, và ta sẽ var\_dump(x), chúng ta sẽ thấy rằng x vẫn có kiểu dữ liệu float, nhưng đây không phải là một số thực sự, vậy làm thế nào để kiểm tra xem nó có phải là vô cùng hay không?

Hàm là is\_infinite và chúng ta nhận được giá trị boolean là true.  Phủ định của điều này sẽ là is\_finite, nói chung là cho bạn biết nếu số không vô cùng. Vì vậy, trong trường hợp này, điều này sẽ là sai.  Nhưng nếu chúng ta đặt x thành một số như 5, thì điều này sẽ là đúng. 

* Sử dụng hàm is\_none để kiểm tra xem một biến có phải là số hay không ?

Để kiểm tra xem biến có phải là một số không, bạn sẽ sử dụng cái gọi là is\_none. Và điều đó sẽ trả về kết quả là true hoặc false. Trong trường hợp này, nó sẽ trả về false.

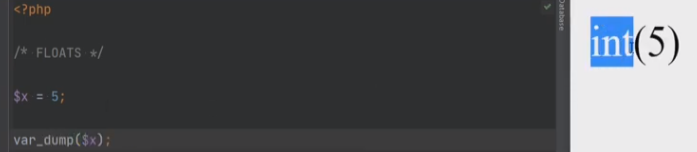
Hãy thử tính log của số âm một và kết quả sẽ là true. Và thực sự, nó trả về true.

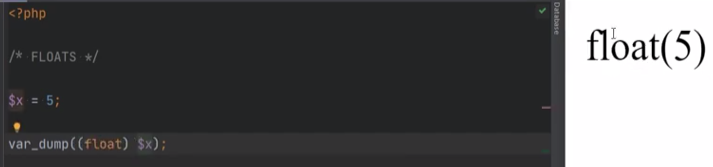


* Giới thiệu về việc ép kiểu (casting)

Cuối cùng, trước khi kết thúc video này, hãy nói về việc ép kiểu (casting).

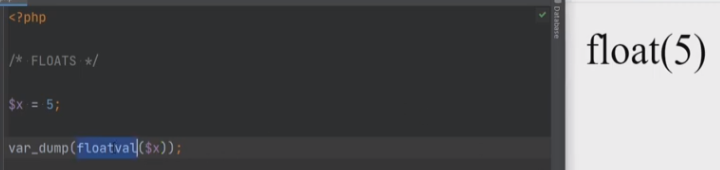
Ví dụ, bạn có một số nguyên như sau và bạn sử dụng var\_dump(x). Kiểu dữ liệu của x sẽ là int.



Cách để chuyển đổi số nguyên này thành số thực (float) là sử dụng toán tử ép kiểu float (float cast). Điều này sẽ chuyển đổi số đó thành kiểu dữ liệu float. 

* Sử dụng hàm floatVal() để chuyển đổi số nguyên thành số thực

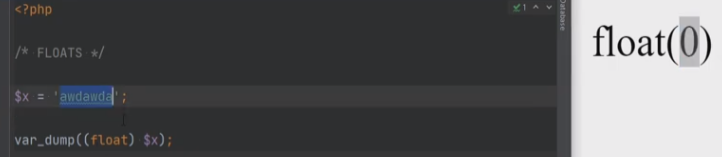
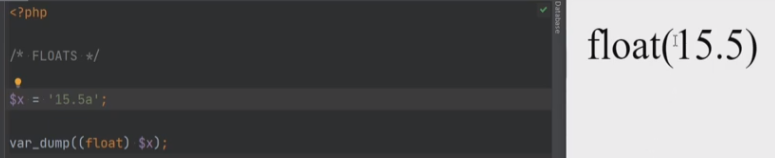
Một cách khác là sử dụng hàm có tên là floatVal(), và điều này cũng có tác dụng tương tự, nhưng cá nhân tôi không thích gọi các hàm không cần thiết vì thực sự không cần gọi hàm này trong trường hợp này.



* Sử dụng từ khóa float để chuyển đổi số nguyên thành số thực và nguyên tắc

Bạn có thể đơn giản chỉ cần sử dụng từ khóa float như sau. Theo ý kiến của tôi, đây là cách tốt hơn. Có một số điều cần lưu ý khi chuyển đổi các kiểu dữ liệu khác sang kiểu float. Khi chuyển đổi một chuỗi thành kiểu float, nếu chuỗi đó có thể được biểu diễn dưới dạng số, có nghĩa là nó là một giá trị số, thì nó sẽ được chuyển đổi thành kiểu float một cách đúng đắn.

Nếu không thể chuyển đổi chuỗi thành số, nó sẽ được chuyển đổi thành số 0.

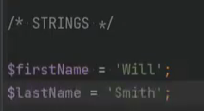
Vì vậy, nếu ta gán một chuỗi không phải là số và chuyển đổi nó thành kiểu float, ta sẽ nhận được kiểu dữ liệu float, nhưng giá trị số sẽ là 0. Nếu chuỗi đó chứa một số như trong ví dụ này, nó sẽ được chuyển đổi thành 15.5. 

1.8 Kiểu dữ liệu chuỗi trong PHP - Cú pháp Heredoc và Nowdoc - Hướng dẫn đầy đủ về PHP 8

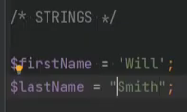
* Giới thiệu về chuỗi (String) trong PHP

String là một chuỗi các ký tự, và chúng ta đã trình bày qua vắn tắt về chuỗi trong các video trước đó, có thể là video đầu tiên hoặc thứ hai của loạt video này, trong đó chúng ta đã thảo luận về cú pháp cơ bản của PHP. Nhưng hãy xem xét nhanh lại và sau đó đi vào một số chi tiết khác về chuỗi mà bạn có thể chưa biết. Khi giá trị được đặt trong dấu nháy đơn hoặc nháy kép, đó là String.

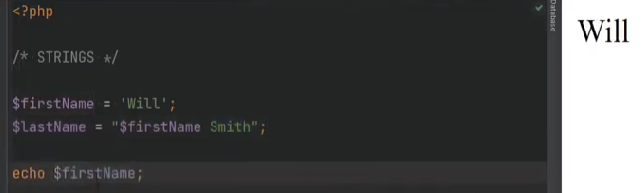
Ví dụ, bạn có thể có tên đầu tiên bằng Will và sau đó tên cuối cùng bằng Smith.

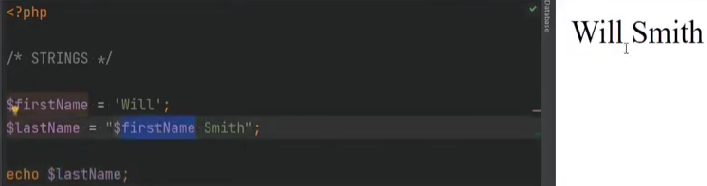


Và đây là các chuỗi vì chúng được đặt trong dấu nháy đơn. Bạn cũng có thể đặt chúng trong dấu nháy kép.

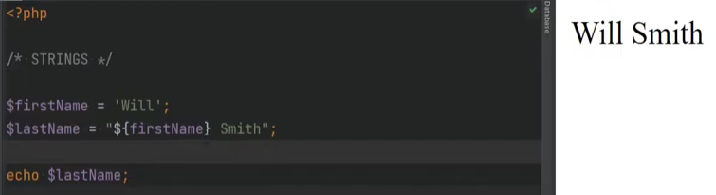
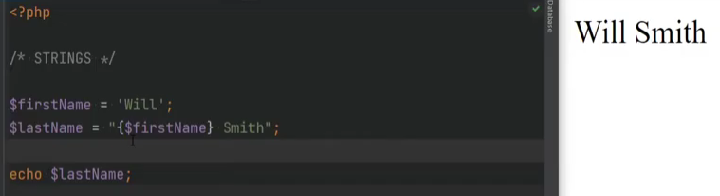


* Sự khác biệt giữa dấu nháy đơn và dấu nháy kép khi sử dụng biến trong chuỗi

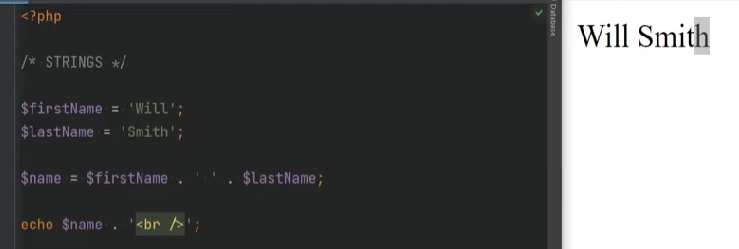
Sự khác biệt chính giữa dấu nháy đơn và dấu nháy kép là trong dấu nháy đơn, bạn không thể sử dụng biến. Và trong dấu nháy kép, bạn có thể sử dụng biến. Vì vậy, trong dấu nháy kép ở đây, bạn có thể nói tên đầu tiên. Và sau đó là khoảng trắng và bất cứ điều gì bạn muốn. Vì vậy, nếu bạn in ra tên đầu tiên, nó sẽ in ra Will. Và nếu bạn in ra tên cuối cùng, nó sẽ in ra tên đầu tiên cùng với tên cuối cùng. Vì vậy, nó sẽ in ra Will Smith.

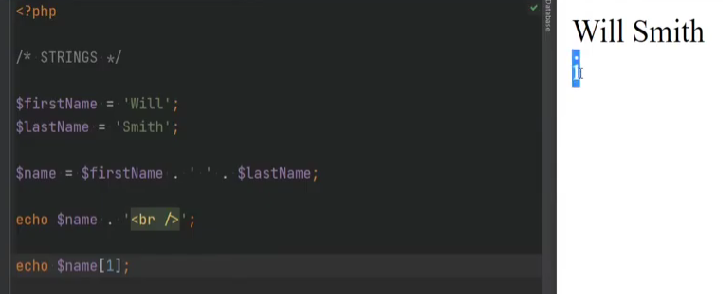
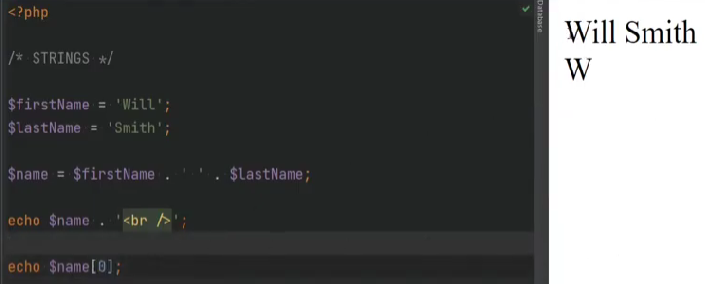


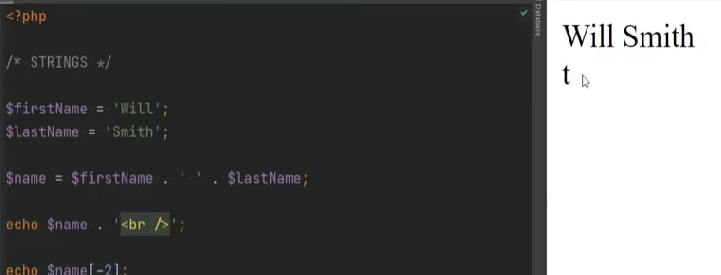
* Sử dụng dấu ngoặc nhọn để bao quanh biến trong chuỗi

Bạn cũng có thể bao quanh tên biến bằng các dấu ngoặc nhọn để đọc dễ hiểu hơn và điều này cũng hoạt động.  Và bạn cũng có thể di chuyển dấu $ này vào bên trong nếu bạn muốn và điều đó cũng hoạt động. 

* Truy cập vào ký tự cụ thể trong chuỗi

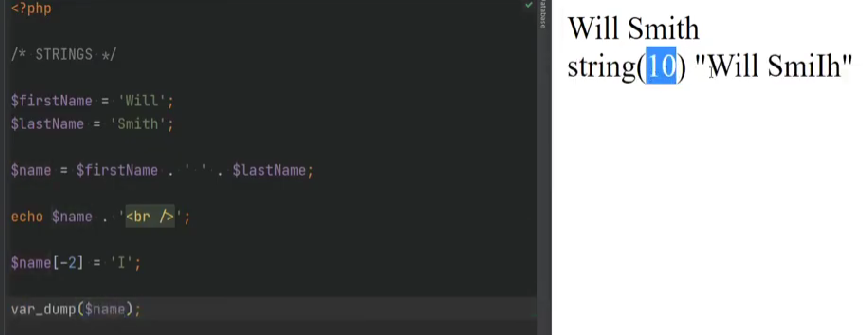
Một trong những điều đó là bạn có thể truy cập vào một ký tự cụ thể trong một chuỗi đã cho. Vì vậy, hãy đổi lại thành Smith và đổi lại thành dấu nháy đơn. Và chúng ta có biến name bằng tên đầu tiên nối với tên cuối cùng và một chuỗi "name" với một dòng mới. Hãy nói rằng chúng ta muốn truy cập vào chữ cái I ở đây. Bạn có thể sử dụng chỉ mục từ 0 để truy cập các ký tự cụ thể trong một chuỗi, cách này giống như truy cập các phần tử trong mảng. Và nếu bạn chưa biết về mảng, đừng lo lắng, chúng ta sẽ trình bày về mảng trong các video sắp tới, nhưng cú pháp vẫn giống nhau. Và nếu bạn chưa biết về mảng, đừng lo lắng. Chúng tôi sẽ bàn về mảng trong các video sắp tới, nhưng cú pháp vẫn giống nhau.

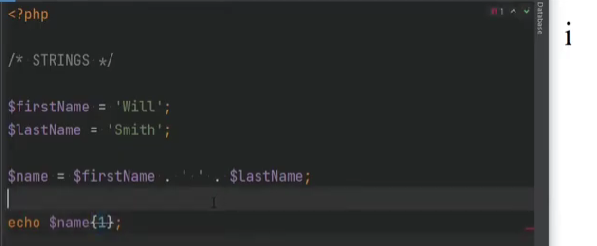
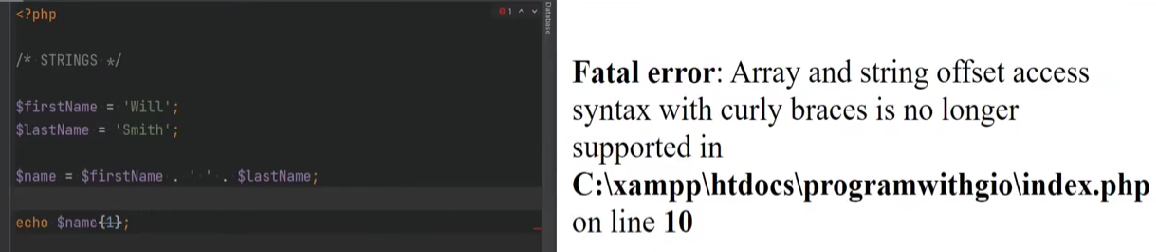
Vì vậy, chữ cái đầu tiên ở đây, chỉ số của nó là 0. Chữ cái tiếp theo là chỉ số 1, chữ cái sau đó là chỉ số 2, và cứ tiếp tục như vậy. Vì vậy, nếu chúng ta muốn in ra chữ cái I, chúng ta sẽ sử dụng lệnh echo name và sau đó đặt dấu ngoặc vuông và số 1 vào trong. Và điều này sẽ in ra chữ cái thứ hai trong chuỗi. Và điều đó sẽ in ra I.  Nếu chúng ta in ra số 0, tức là chữ cái đầu tiên, nó sẽ in ra W.  Bây giờ, nếu bạn muốn truy cập từ phía sau thay vì từ phía trước, bạn có thể sử dụng số âm. Vì vậy, bạn có thể sử dụng số âm hai, và điều này sẽ in ra chữ cái thứ hai từ phía sau. Vì vậy, trong trường hợp này, là chữ T.

 Và nếu chúng ta làm mới, nó sẽ in ra t. Hãy thay đổi chữ cái I này thành chữ cái I in hoa. Và chúng ta có thể làm điều đó bằng cách sử dụng dấu ngoặc vuông cách nhau như vậy. Vì vậy, name[1] = I. Và nếu chúng ta in ra name bây giờ, chúng ta sẽ thấy rằng nó đã được thay đổi ở đây.

 Và chúng ta có thể làm điều tương tự với các số âm ở đây. Vì vậy, chúng ta có thể thay đổi chữ cái thứ hai từ phía sau, tức là chữ T. Và bây giờ chúng ta sẽ thay đổi chữ T thành chữ I in hoa, và điều đó hoạt động.

 Hãy sử dụng var\_dump(name) ở đây. Và chúng tôi thấy rằng chúng ta có tổng cộng 10 chữ cái, điều đó có nghĩa là chỉ số của chữ cái cuối cùng là 9.

 Nhưng nếu chúng ta đặt một chữ cái tại chỉ số 15, ví dụ, nó chỉ đơn giản là thêm các khoảng trống vào chuỗi. Vì vậy, nếu chúng ta làm mới điều này. Bây giờ chúng ta có những khoảng trống trống này ở đây và độ dài của chuỗi là 16 thay vì 10.

Vì vậy, hãy nhớ khi bạn cập nhật một chuỗi như thế này. Một điều đáng nhắc đến khác là trước đây có một cách khác để truy cập vào các ký tự chuỗi và các phần tử mảng bằng cách sử dụng cú pháp dấu ngoặc nhọn thay vì dấu ngoặc vuông. Thay vì sử dụng câu lệnh echo name[1], bạn có thể làm điều tương tự bằng cú pháp dấu ngoặc nhọn, và điều này đã hoạt động trước PHP 8. Tuy nhiên, trong PHP 8, nó sẽ gây ra lỗi vì cú pháp này đã bị lược bỏ từ phiên bản PHP 7.4 trở đi. 

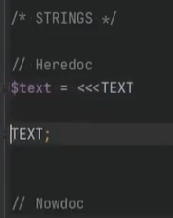
* Giới thiệu về Heredoc và Nowdoc

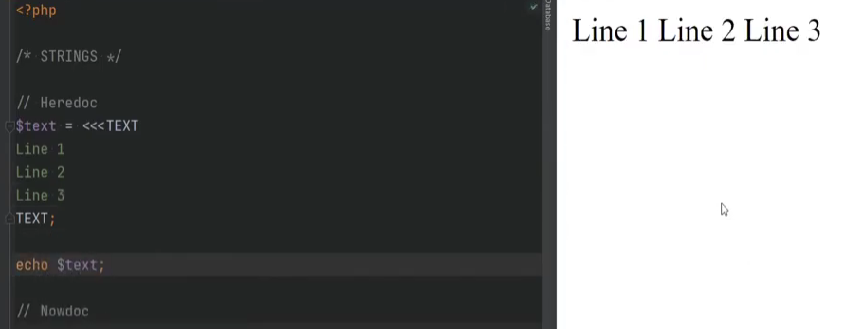
Cho đến nay, bạn đã thấy cú pháp dấu nháy đơn hoặc dấu nháy kép để biểu diễn chuỗi. Nhưng thực tế, còn hai cú pháp khác gọi là Heredoc và Nowdoc.

Heredoc và Nowdoc là cách để xử lý chuỗi dài và đa dòng có thể chứa các biểu thức phức tạp và cũng có thể chứa cả dấu nháy đơn và dấu nháy kép mà không cần phải thoát chúng.

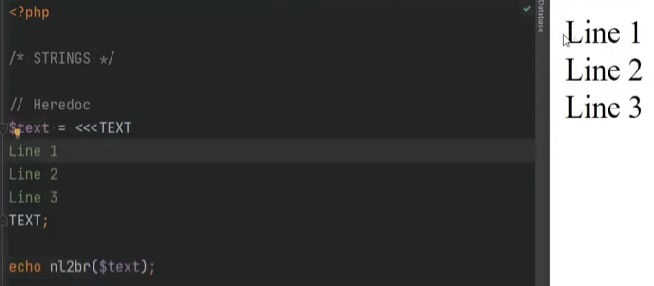
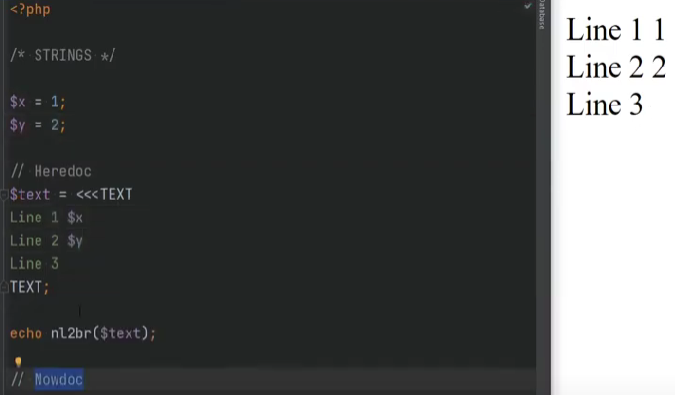
* Cú pháp và sự khác biệt giữa Heredoc và Nowdoc

Sự khác biệt chính giữa hai cú pháp này là Heredoc xem chuỗi như được bao bọc trong dấu nháy kép, trong khi Nowdoc không xem chuỗi như vậy. Chuỗi giống như được bao trong dấu nháy đơn, có nghĩa là trong Heredoc, bạn có thể có biến, trong khi trong Nowdoc, bạn không thể có biến. Và cú pháp gần như giống nhau. Trong trường hợp này, bạn tạo một biến, hãy gọi là text, sau đó có ba dấu nháy nhỏ hơn và sau đó là một định danh cho text của bạn. Trong trường hợp này, hãy gọi nó là text như vậy. Sau đó, bạn kết thúc nó bằng định danh của bạn và một dấu chấm phẩy.

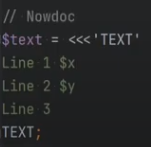


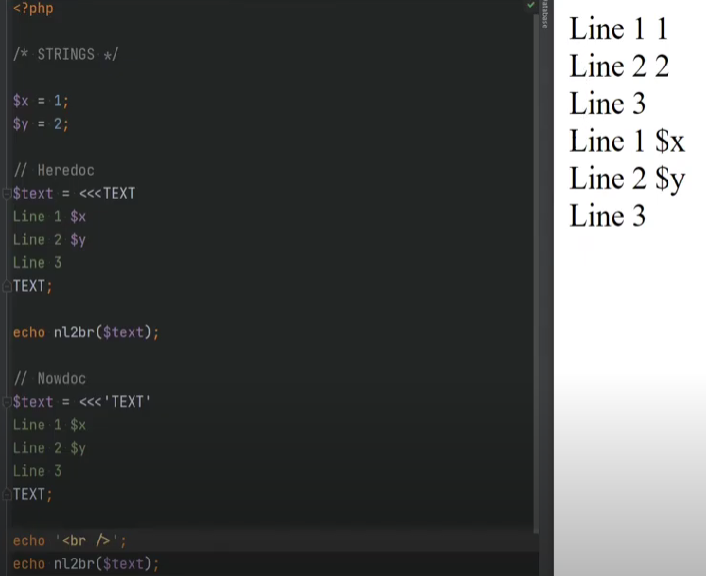
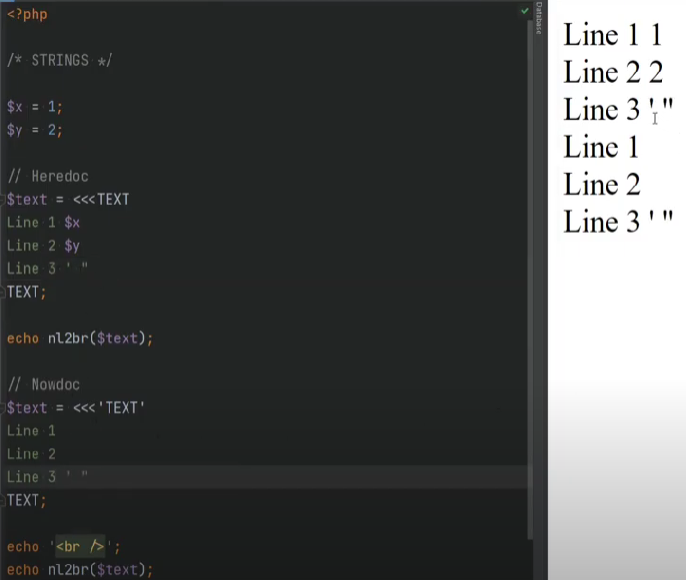
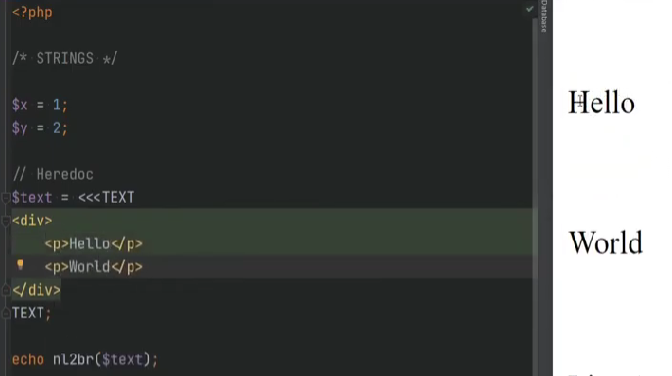
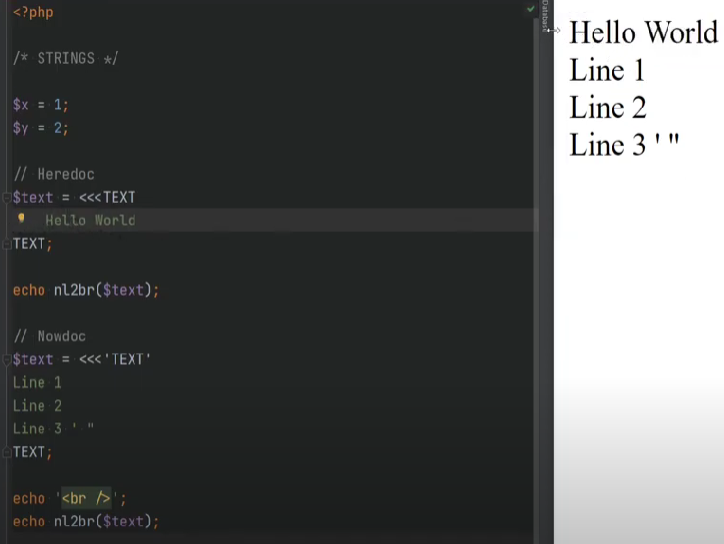
Và sau đó, bất cứ điều gì ở đây chỉ là nội dung của bạn. Vì vậy, bạn có thể có dòng 1, dòng 2, dòng 3. Và chúng ta có thể sử dụng câu lệnh echo để hiển thị nội dung của text. Trong trình duyệt, điều này sẽ hiển thị tất cả các dòng trong một dòng duy nhất vì đó thực sự là các ký tự xuống dòng và trình duyệt không đọc chúng một cách chính xác. 

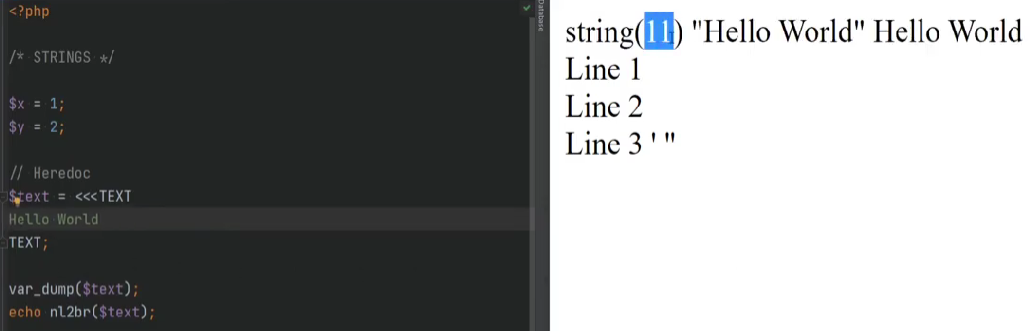
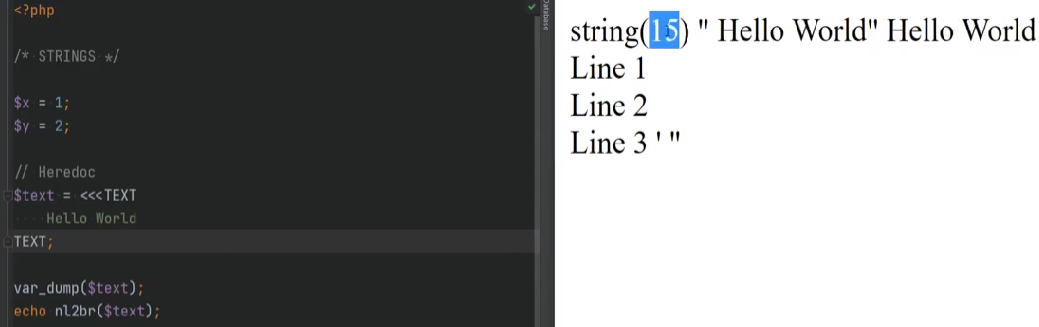
Chúng ta có thể sửa lỗi này bằng cách sử dụng hàm nl2br. Nếu chúng ta làm mới trang, các dòng mới sẽ được hiển thị đúng. Như bạn có thể thấy, điều này rất hữu ích.

 Vì vậy, bạn có thể có một biến ở đây được gọi là x bằng 1, y bằng 2 và bạn có thể sử dụng chúng trong đoạn mã này, x, y, và điều này sẽ hoạt động.  Cú pháp cho NowDoc giống với HereDoc, khác biệt duy nhất là định danh cần được bao bọc bằng dấu nháy đơn và khi bạn làm như vậy, các biến sẽ không còn được in ra nữa, điều đó có nghĩa là đoạn mã này được xem như một dấu nháy đơn.

* Ứng dụng của Heredoc và Nowdoc trong việc tạo mã HTML động



Nếu chúng ta tiếp tục sử dụng câu lệnh echo ở đây, hãy thêm một dòng xuống dòng khác ở đây, chúng ta sẽ thấy rằng trong phần này, sử dụng nowdoc, các biến không được in ra đúng cách. Bây giờ bạn có thể đặt câu hỏi về việc sử dụng cho điều này là gì?  Trong thực tế, có rất nhiều trường hợp sử dụng cho điều này, một trong số đó là khi bạn có chuỗi đa dòng với các biểu thức phức tạp, heredoc hoặc nowdoc thực sự rất dễ đọc và tốt hơn rất nhiều so với việc sử dụng nối chuỗi hoặc nhiều dấu nháy và bạn cần phải thoát dấu nháy và các ký tự tương tự. Như bạn đã nhận thấy ở đây, bạn không cần phải thoát bất cứ điều gì. Bạn có thể có cả dấu nháy đơn và dấu nháy kép trong cả heredoc và nowdoc. Vì vậy, nếu tôi làm mới trang, cả hai đều xuất hiện mà không cần thoát chúng.  Một trường hợp sử dụng khác là bạn có thể tạo ra một số mã HTML động và điều này rất hữu ích. Bạn thực sự có thể có các thẻ HTML đầy đủ ở đây như thế này. Vì vậy, bạn có thể tạo một đoạn văn bản "hello world" và nó sẽ in ra "hello world".  Chỉ cần lưu ý rằng khi sử dụng Heredoc hoặc Nowdoc, bất kỳ dấu cách trong đó đều được hiển thị chính xác như vậy. Ví dụ, khi chúng ta có "hello world" ở đây, nó in ra "hello world".  Nhưng nếu bạn thêm một tab hoặc một dấu cách nào đó ở đây, nó sẽ thêm dấu cách đó vào. 

Bạn có thể không nhìn thấy nó trên trình duyệt, nhưng nó thực sự có mặt. Chúng ta có thể sử dụng câu lệnh var\_dump(text) và hãy xóa dấu cách tạm thời. Chúng ta sẽ thấy rằng nó có 11 ký tự.  Bây giờ, hãy thêm tab trở lại và chúng ta sẽ thấy nó thay đổi thành 15.  Điều đó có nghĩa là có bốn dấu cách ở đầu. Và nếu chúng ta xóa và chỉ thêm một dấu cách, chúng ta sẽ có 12 ký tự.

