جامعة طرابلس
كلية تقنية المعلومـــــات
قسم نظم المعلومات

المقرر الدراسي:

تطوير التطبيقات

Application development

ITIS311

المحاضرة (4)

اعداد: أ. فاطمة بشير القاضي
F.alqadhi@uot.edu.ly

حلقات الدوران في PHP

هناك 4 أنواع من حلقات الدوران

حلقة الدوران for: تنفيذ مجموعة من الاوامر لعدد محدد من المرات .

حلقة الدوران while: تنفيذ مجموعة من الاوامر عند تحقق الشرط المحدد .

حلقة الدوران do...while: تنفيذ مجموعة من الاوامر لمرة واحدة ثم تكرار الحلقة طالما يتحقق الشرط المحدد.

حلقة الدوران foreach (حلقة دوران التعامل مع المصفوفات): تنفيذ مجموعة من الاوامر لكل عنصر في المصفوفة .

حلقة الدوران For

تستخدم الحلقة For عند معرفة عدد المرات التي يجب أن تنفذ بها الاوامر.

```
for(/*بداية الحلقة*/;/*الشرط */;/*معامل الزياد ة أو النقصان*/)
{
/*
الاوامر البرمجية المراد تكرارها عدد من المرات
*/
}
```

Init بداية الحلقة: تستخدم لضبط القيمة المبدئية للعداد.

Condition الشرط: هو الشرط الذي سيرفق في الحلقة في حال كان الشرط true فإن الحلقة ستكمل الدوران ولكن في حال كان الشرط false ستتوقف الحلقة.

Increment معامل الزيادة أو النقصان: يستخدم لزيادة أو نقصان العداد.

تابع حلقة الدوران For

ملاحظة : في حالة عدم وجود أقواس يتم تنفيذ الامر البرمجي بعد For وصولاً لنهاية الامر البرمجي المنتهي بالفاصلة المنقوطة ;

Ex1:

```
<?php
 for ($i=1; $i<=5; $i++)
{
   echo  " الرقم هو " . " " .$i ;
   echo "<br />";
}
 ?>
```

Ex2:

```
<?php
 for ($i=10; $i > 5; $i--)
echo  " الرقم هو " . " " .$i . "<br />";
?>
```

تابع حلقة الدوران For

الكلمة المحجوزة continue:

تستخدم لتخطي دورة معينة والانتقال لتنفيذ الدورة التي تليها.

```
<?php
for($i=0 ; $i<10 ; $i++)
{
    if($i == 5)
    continue;
    echo '<h3>$i='.$i.'</h3>';
}
?>
```

تابع حلقة الدوران For

الكلمة المحجوزة break:

تستخدم للخروج من الحلقة بشكل نهائي .

```
<?php
for($i=0 ; $i<10 ; $i++)
{
    if($i == 5)
    break;
    echo '<h3>$i='.$i.'</h3>';
}
?>
```

حلقة الدوران while

تستخدم حلقة الدوران while لتنفيذ مجموعة من الاوامر عند تحقق شرط معين، وفي حالة عدم تحققه لا يتم الدخول للحلقة.

```
<?php
While (/*الشرط*/)
{
/*
الاوامر المراد تكرارها
*/
}
?>
```

تابع حلقة الدوران while

```
<?php
$count = 0;
while(10)
{
   echo "<h3> Hi </h3>";
}
while('user')
{
echo "<h3> Hi </h3>";
}
while($count != 10)
{
echo "<h3> Hi </h3>";
}
?>
```

تابع حلقة الدوران while

جميع الحلقات السابقة حلقات غير منتهيه وتسبب تجمد المتصفح والضغط على الخادم، والسبب أن الشرط محقق دائماً كما نعلم .أمثلة على حلقات صحيحة ومنتهية :

```
<?php
$count = 1;
While ($count <= 10)
{
echo "<h3> Hi </h3>";
$count++;
}
While (true)
{
echo "<h3> YES </h3>";
 if ($count++ == 20) break;
}
?>
```

حلقة الدوران Do while

تستخدم حلقة الدوران Do while لتنفيذ مجموعة من الاوامر لمرة واحدة ثم تكرار الحلقة طالما تحقق الشرط .

```
<?php
 do
{
/*
 الاوامر المراد تكرارها
*/
}      while(/*الشرط*/);
?>
```

تابع حلقة الدوران Do while

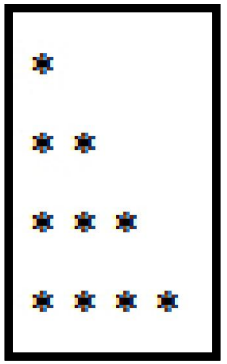
```
<?php
 do
{
echo "<h3>Hi</h3>";
} while(false);
$count = 0;
do
{
echo '<h3>$count = '.++$count.'</h3>';
} while($count < 10);
?>
```

ملاحظة: في كل حلقات الدوران السابقة يمكن استخدام continue لتخطي دورة واحدة أو الخروج نهائياً من الحلقة باستخدام break.

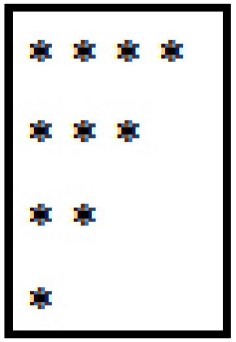
الحلقات التكرارية المتداخلة

الحلقة المتداخلة هي حلقة داخل حلقة، أوهي حلقة داخلية داخل جسم الحلقة الخارجية.

```
Ex1: <?php
for ($i = 1; $i < 5; $i++) {
       for ($j = 1; $j <= $i; $j++) {
            echo " * ";
      }
    echo '<br />';
} ?>
```



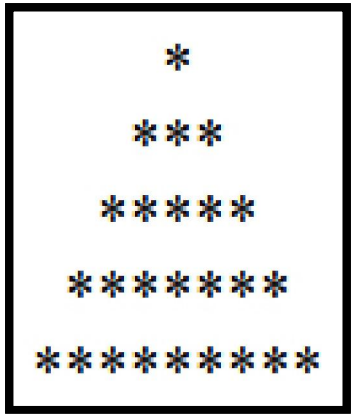
```
Ex2: <?php
 for ($i = 1; $i < 5; $i++) {
    for ($j = $i; $j < 5; $j++) {
        echo " * ";
     }
    echo '<br />';
 } ?>
```



تابع الحلقات التكرارية المتداخلة

Ex3:

```
<?php
    for($i = 0; $i < 5; $i++) {
        // print spaces
        for($j = 0; $j < 5 - $i - 1; $j++) {
            echo "  ";
        }
        // print stars
        for($k = 0; $k < 2 * $i + 1; $k++) {
            echo "*";
        }
        echo "<br>";
    }
?>
```



تابع الحلقات التكرارية المتداخلة

حلقات While متداخلة

Ex4:

```
<?php
$a = 3;
while ($a <= 4)            //before check if conditon is true then executed internal parts with check internal while loop conditions
//if wrong then exit
{
echo "this is example $a <br>";
 ++$a;
 $b = 4;
while ($b <= 5) {
 echo "val $b <br>";
 ++$b;
    }
 } ?>
```

المصفوفات Arrays

ماهي المصفوفة Array؟

المصفوفة هي متغير خاص حيث يمكنه تخزين قيمة أو أكثر تحت نفس المسمى.

اذا كان لديك قائمة من العناصر (كتب على سبيل المثال) عندها يمكنك تخزينهم في متغير واحد .

```
$book1= " كتاب البرمجة ";
$book2= " كتاب الحاسوب ";
$book3= " قصص المغامرات ";
```

ولكن ماذا لو أردت أن تجلب كتاب معين من بين هذه الكتب؟ وماذا لو كان لديك أكثر من 300 كتاب ليس فقط 3 ؟ ماذا ينبغي أن تفعل ؟

أفضل حل هو إنشاء مصفوفة array()، يمكن للمصفوفة أن تحمل جميع القيم التي لديك وأيضاً جميع المتغيرات التي لديك بمتغير واحد فقط ويمكنك استدعاء القيمة التي تريد من خلال ذكر اسمها و ترتيبها فقط.

يحتوي كل عنصر في المصفوفة على ID خاص به وبذلك يمكن استدعائه بكل سهولة.

يمكن ان تحتوي المصفوفة الواحدة في PHP على مجموعة بيانات متعددة الانواع لأنه كما في المتغيرات لا يتم تحديد نوع المصفوفة عند التعريف عنها.

تابع المصفوفات Arrays

يوجد 3 أنواع للمصفوفات في لغة PHP:

- ✓ **المصفوفة الرقمية** numeric arrays: تحتوي على فهرسة رقمية .
- ✓ **المصفوفة المترابطة** Associative arrays : تتكون من مجموعة ID وكل ID يحمل قيمته الخاصة .
- ✓ **المصفوفة متعددة الابعاد** Multidimensional arrays : عبارة عن مصفوفة تحتوي على مصفوفة او أكثر من مصفوفة اخرى.

في PHP لتخزين قيم ما على شكل مصفوفة عليك فقط أن تضع الاقواس المربعة [] بعد اسم المتغير، وتقوم بإسناد القيم للمصفوفة كالتالي:

```
<?php
$myArr[] = 10;          //key = 0 , value = 10
$myArr[] = 12.16;       //key = 1 , value = 12.16
$myArr[] = true;        //key = 2 , value = true
$myArr[] = "username";          //key = 3 , value = "username"
$myArr[] = 'password';          //key = 4 , value = 'password‘ ?>
```

المصفوفة الرقمية

تخزن المصفوفة الرقمية كل عنصر من عناصر المصفوفة برقم مفهرس .

هناك طريقتين لإنشاء المصفوفة الرقمية:

1. في هذا المثال تم تعيين الفهرسة تلقائياً (تبدأ الفهرسة من الرقم 0) .

```
$ myBooks = array("كتاب البرمجة"،"كتاب الحاسوب"، قصص المغامرات");
```

2. في المثال التالي سيتم إنشاء مصفوفة وتعيين الفهرسة يدوياً .

```
$myBooks [0]= " كتاب البرمجة ";
$myBooks [1]= " كتاب الحاسوب ";
$myBooks [2]= " قصص المغامرات ";
```

Ex:

```
<?php
$myBooks [0]= " كتاب البرمجة ";
$myBooks [1]= " كتاب الحاسوب ";
$myBooks [2]= " قصص المغامرات ";
echo  "يعتبر ". $myBooks[0] . " و ". $myBooks[1] . " من أفضل الكتب العلمية";   ?>
```

المصفوفة المترابطة

تستخدم المصفوفة المترابطة مسميات للمفاتيح named keys عند تخزين القيم.

هناك طريقتين لإنشاء المصفوفة المترابطة:

1. في هذا المثال سيتم تحديد قيم محددة اي سيتم تحديد اسم الشخص مع عمره .

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

2. يعتبر هذا المثال مماثل للمثال في 1 ولكن يظهر طريقة أخرى لإنشاء المصفوفة المترابطة.

```
$ages['Peter'] = "32"; $ages['Quagmire'] = "30"; $ages['Joe'] = "34";
```

Ex:

```
<?php
 $ages['Peter'] = "32";
 $ages['Quagmire'] = "30";
 $ages['Joe'] = "34";
 echo "Peter is " . $ages['Peter'] . " years old."; ?>
```

المصفوفة متعددة الابعاد

نستطيع تبسيط هذا النوع بأنه عبارة عن مصفوفة تحتوي على مصفوفة او اكثر.

ملاحظة: PHP تستطيع فهم والتعامل مع مصفوفات من النوع Multidimensional arrays، التي تحتوي على مصفوفة ، او مصفوفتين ، او ثلاث مصفوفات ، و حتى اربع ، و خمس ابعاد من المصفوفات المخزنة ، لكن المصفوفات التي تحتوي على اكثر من 3 ابعاد من المصفوفات المخزنة صعبة في التعامل من قبل اغلب المطورين.

هناك طريقتين لإنشاء المصفوفة متعددة الابعاد:

1. اسناد القيم دفعة واحد ة عند تعريف المتغير

```
$myArr = array(
array('username', "password", 10),
array(12, 45.99, true)
);
```

تابع المصفوفة متعددة الابعاد

2. تمثيل المصفوفات متعدد ة الابعاد على أنها مصفوفات أحادية متداخلة

```
$myArr = Array (
 [Griffin] => Array ( [0] => Peter [1] => Lois [2] => Megan )
 [Quagmire] => Array ( [0] => Glenn )
 [Brown] => Array ( [0] => Cleveland [1] => Loretta [2] => Junior ));
```

Ex:

```
<?php
  echo "Is " . $families['Griffin'][2] .
  " a part of the Griffin family?";
 ?>
```

حلقة الدوران Foreach

حلقة foreach للدوران على عناصر المصفوفة من أفضل الطرق للدوران على عناصر المصفوفة وبالاخص المصفوفات المترابطة، ويمكن من خلالها إستخراج القيمة أو القيمة و المعرف (المفتاح)والشكل العام لها.

```
foreach ($array as $key => $value){
//$key هو مفتاح المصفوفة
//$value هي القيمة المرتبطة بالمفتاح }
```

والمثال التالي يواضح فكرة عملها :

```
<?php   $myArr = array('name' => 'username 1', 'city' => 'luxor', 'phone‘ => 125668522);
foreach($myArr as $value)
echo "<h3>$value</h3>";
 foreach($myArr as $key=>$value)
echo "<h3>$key : $value</h3>";
?>
```

دوال التعامل مع المصفوفات

1.الدالة count

تستخدم لمعرفة عدد عناصر المصفوفة, والتي تقبل وسيطا واحدا وهو المصفوفة المراد معرفة عدد عناصرها، وتعيد عدد عناصر المصفوفة.

```
<?php
$myArr = array(10, 12.16, true, "username", 'password');
for($i = 0; $i < count($myArr); $i++)
echo "<h3>{$myArr[$i]} </h3>";    ?>
```

ملاحظة : تطبع علامة التنصيص المزدوجة قيمة المتغيرات اما لوحدها او باستخدام اقواس المجموعة { }، أما علامة التنصيص المفردة تطبع اسم المتغير فقط.

```
<?php
 $name = " PHP Course " ;
echo "the site name is $name <br> " ;
echo ‘the site name is $name <br> ’;
echo "the site name is {$name} <br> ";   ?>
```

تابع دوال التعامل مع المصفوفات

2. دوال الطباعة

هناك دوال مهمتها عرض محتويات وبيانات المتغيرات و المصفوفات والكائنات وهي var_dump و print_r و var_export سنستخدمها لطباعة محتويات المصفوفة من القيم والـ key لكل قيمة , وتقبل - هذه الدوال - وسيطا واحداً هو المصفوفة المراد طباعتها ,كما في المثال التالي :

```
<?php
$myArr = array('name' => 'username 1', 'city' => 'luxor', 'phone‘ => 125668522);
echo var_dumpt($myArr) . <br />;
echo print ($myArr) . <br />;
echo var_export($myArr) . <br />;
?>
```

تابع دوال التعامل مع المصفوفات

3. الدالة explode

تقوم هذه الدالة بتقطيع نص وتحويله الى مصفوفة حيث تقبل وسيطين اجباريين الوسيط الاول هو "الفاصل "الذي عنده يتم اقتطاع الجملة و الوسيط الثاني هو النص.

مثال: بفرض اننا نريد ان نجعل كل كلمة في جملة معينة عنصرا من عناصر مصفوفة وبالتالي يكون الفاصل هو "الفراغ ":

```
<?php
$string = 'this is a sting';
$array = explode(' ', $string); print_r($array);
?>
```

ملاحظة: تستخدم هذه الدالة بكثر ة عند القراء ة من الملفات النصية

تابع دوال التعامل مع المصفوفات

4. الدالة implode

تقوم هذه الدالة -تقريباً- بعكس عمل الدالة explode أي انها تقوم بتحويل عناصر مصفوفة الى نص يفصل بينها "فاصل" ، حيث الوسيط الاول هو الفاصل و الوسيط الثاني هو المصفوفة المراد تحويل جميع عناصرها الى سلسلة نصية.

```
<?php
$array = array(10, 12.16, true, "username", 'password');
$string = implode(' -- ', $array);
echo $string; ?>
```

5. الدالة is_array

تقوم هذه الدالة بالتحقق من ان الوسيط الممرر لها هو مصفوفة وذلك بإعادة القيمة true او false :

```
<?php
$array = array(10, 12.16, true, "username", 'password');
echo is_array($array);  ?>
```

تابع دوال التعامل مع المصفوفات

6. دالة إضافة قيمة الى المصفوفة array_push

يمكن اضافة عنصر جديد للمصفوفة بواسطة القوسين [] كالتالي :

```
<?php
$array = array('sy', 'eg', 'lb');
$array[] = 'sa';
echo '<br>the array after adding sa is :<br>';
print_r ($array);   ?>
```

او باستخدام الدالة array_push حيث تقبل وسيطين الاول هو المصفوفة الهدف والثاني القيمة المراد اضافتها

```
<?php
$array = array('sy', 'eg', 'lb');
array_push ($array, 'sa');
echo '<br> the array after adding sa is :<br>';
print_r ($array);    ?>
```

تابع دوال التعامل مع المصفوفات

7. البحث داخل المصفوفات in_array

نستخدم الدالة in_array للبحث داخل المصفوفة عن قيمة معينة ,هذه الدالة تعيد true في حال نجاحها.

```
<?php
$array = array('sy', 'eg', 'lb', 'sa');
 if(in_array('sa', $array) == true)
  echo ' sa is found in $array array <br>';
else
  echo ' fr is NOT found in $array array <br>';    ?>
```

8. قلب مصفوفة array_reverse

تستخدم الدالة array_reverse لقلب ترتيب مصفوفة اي جعل اول عنصر اخر عنصر وهكذا.

```
<?php
$array = array('1', '2', '3', '4');
$new_array = array_reverse($array);
print_r($new_array);?>
```

تابع دوال التعامل مع المصفوفات

9. الدالة array_unique

تقوم الدالة array_unique بإزالة أي قيمة تتكرر في المصفوفة, حيث تعيد مصفوفة جديد ة بدون أي عناصر مكررة.

```
<?php
$array = array('sy', 'eg', 'lb', 'sy', 'lb', 'sa');
$new_array = array_unique($array);
echo 'the first array is : ';
print_r($array);
echo '<br> the "unique" one : ';
print_r($new_array);
?>
```

تابع دوال التعامل مع المصفوفات

10. ترتيب عناصر المصفوفة sort ,asort

تقوم الدالة sort بترتيب عناصر مصفوفة تصاعدياً،ولا تعيد هذه الدالة أي قيمة ,أي تقوم بتعديل المصفوفة مباشرة حيث تستقبل وسيط وحيد وهو المصفوفة المراد ترتيب عناصرها.

```
<?php
$array = array(123, 1, 12, 'name' => 'sy', 'eg');
print_r($array);
sort($array);
echo '<br>';
 print_r($array);
 ?>
```

لاحظ أن المصفوفة المرتبة لا تحتفظ بمفاتيح المصفوفة الاصلية، وللاحتفاظ بها نستخدم الدالة asort

تابع دوال التعامل مع المصفوفات

تقوم الدالة asort بنفس عمل الدالة sort لكنها تحتفظ بقيم المفاتيح أو المعرفات .

```
<?php
$array = array(123, 1, 12, 'name' => 'sy', 'eg');
 print_r($array);
asort($array); echo '<br>';
 print_r($array);
?>
```