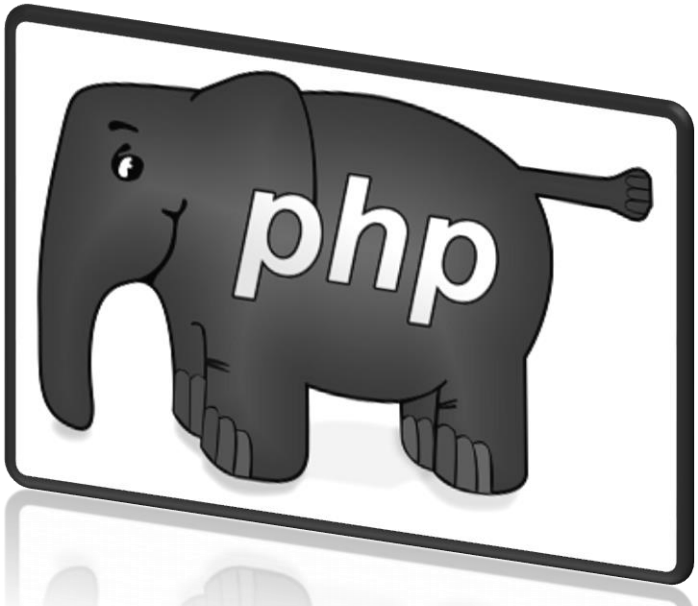
Langage PHP

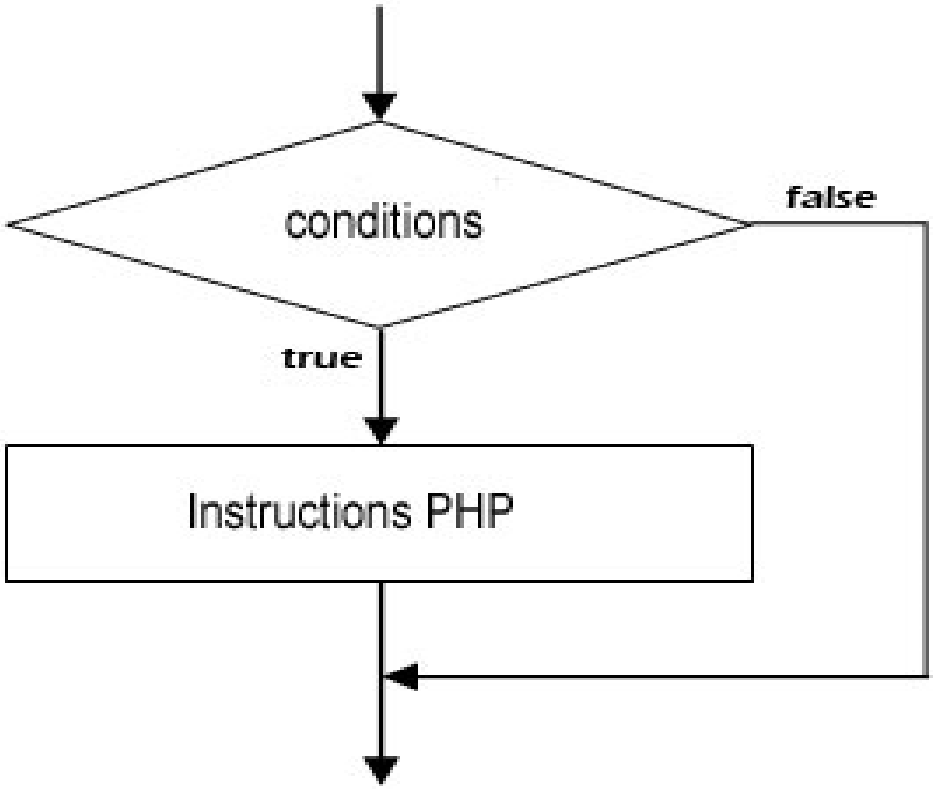


FAQIR Nada

(nadafaqir@gmail.com

Chapitre 5

- Les instructions de contrôle



Introduction

- On retrouve dans PHP la plupart des instructions de contrôle des scripts. Indispensables à la gestion du déroulement d'un algorithme quelconque, ces instructions sont présentes dans tous les langages. PHP utilise une syntaxe très proche de celle du langage C.
- La version 5 de PHP a vu l'apparition de nouvelles instructions dédiées à la gestion des exceptions, comme try…catch ou throw, qui lui faisaient défaut jusqu'à présent.

Les instructions conditionnelles

- Comme tout langage, PHP dispose d'instructions conditionnelles qui permettent d'orienter le déroulement d'un script en fonction de la valeur de données.

Syntaxe de base : Les instructions conditionnelles

- L'instruction if
  - if (condition réalisée) { liste d'instructions }
- L'instruction if ... Else
  - if (condition réalisée) {liste d'instructions} else { autre série d'instructions }
- L'instruction if ... elseif ... Else
  - if (condition réalisée) {liste d'instructions} elseif (autre condition) {autre série d'instructions } else (dernière condition réalisée) { série d'instructions }
- Opérateur ternaire
  - (condition) ? instruction si vrai : instruction si faux

L'instruction if

- L'instruction if est la plus simple et la plus utilisée des instructions conditionnelles.
- Présente dans tous les langages de programmation, elle est essentielle en ce qu'elle permet d'orienter l'exécution du script en fonction de la valeur booléenne d'une expression.
- Sa syntaxe est la suivante :
  - if (condition réalisée) { instruction }

- L'instruction if peut être suivie d'un bloc d'instructions délimité par des parenthèses qui sera entièrement exécuté dans les mêmes conditions :

```
if(expression)
{
//bloc de code
}
```

- La rédaction de l'expression est importante. Elle peut devenir complexe lorsqu'elle comprend des opérateurs logiques associant ses différents composants.
- Dans le code suivant :

```
<?php
$a=6;
if(is_integer($a) && ($a<10 && $a>5) && ($a%2==0) ) {echo "Conditions satisfaites";}
?>
```

L'instruction if...else

- L'instruction if...else permet de traiter le cas où l'expression conditionnelle est vraie et en même temps d'écrire un traitement de rechange quand elle est évaluée à FALSE, ce que ne permet pas une instruction if seule. L'instruction ou le bloc qui suit else est alors le seul à être exécuté. L'exécution continue ensuite normalement après le bloc else.

Exemple

- L'exemple 5-1 suivant calcule le prix net après une remise variable en fonction du montant des achats selon les critères suivants :
  - Si le prix total est supérieur à 100 euros, la remise est de 10 %. Cette condition est traitée par l'instruction if (repère →1).
  - Pour les montants inférieurs ou égaux à 100 euros, la remise est de 5 %. Cette condition est traitée par l'instruction else (repère →2).

Exemple 5-1

```
<?php
$prix=55;
if($prix>100) ←❶
{
  echo "<b>Pour un montant d'achat de $prix &euro;, la remise est de 10 % </b>
  ➥<br />";
  echo "Le prix net est de ",$prix*0.90;
}
else ←❷
{
  echo "<b>Pour un montant d'achat de $prix &euro;, la remise est de 5 %</b><br />";
  echo "<h3>Le prix net est de ",$prix*0.95,"</h3>";
}
?>
```

L'instruction if...else

- Le bloc qui suit les instructions if ou else peut contenir toutes sortes d'instructions, y compris d'autres instructions if...else. Nous obtenons dans ce cas une syntaxe plus complexe, de la forme :

```
if(expression1)
{//Bloc 1}
elseif(expression2)
{//Bloc 2}
else
{//Bloc 3}
```

Exemple

- Dans l'exemple 5-2, vous voulez afficher le montant d'une remise calculée selon les modalités suivantes :
  - Si vous achetez un PC de plus de 1 000 euros, la remise est de 15 %.
  - Pour un PC de 1 000 euros et moins, la remise est de 10 %.
  - Pour les livres, la remise est de 5 %.
  - Pour tous les autres articles, la remise est de 2 %.

```php
<?php
// ********************if...elseif...else************** **
$cat="PC";
$prix=900;
if($cat=="PC") ←❶
{
  if($prix >= 1000) ←❹
  {
     echo "<b>Pour l'achat d'un PC d'un montant de $prix &euro;, la remise est de
     ➥15 %</b><br />";
     echo "<h3> Le prix net est de : ".$prix*0.85, "&euro; </h3>";
  }
  else ←❺
  {
    echo "<b>Pour l'achat d'un PC d'un montant de $prix &euro;, la remise est de
    ➥10 %</b><br />";
    echo "<h3> Le prix net est de : ".$prix*0.90, "&euro; </h3>";
  }
}
elseif($cat=="Livres") ←❷
{
  echo "<b>Pour l'achat de livres la remise est de 5 %</ b><br />";
  echo "<h3> Le prix net est de : ".$prix*0.95, "&euro; </h3>";
}
else ←❸
{
  echo"<b>Pour les autres achats la remise est de 2 %</ b><br />";
  echo "<h3> Le prix net est de : ".$prix*0.98, "&euro; </h3>";
}
?>
```

Exemple 5-2. Les instructions if imbriquées

- La première instruction if…elseif…else (repères 1, 2 et 3) contrôle la catégorie du produit. La deuxième instruction if…else détermine le montant de la remise si le produit est un PC (repères 4 et 5). L'indentation du code permet une lecture plus facile. Faute d'une telle indentation, il est difficile de distinguer à quelle instruction if se rapporte une instruction else.

L'opérateur ?

- L'opérateur ? permet de remplacer avantageusement une instruction if...else en évaluant une expression et en attribuant à une variable une première valeur si la condition est vraie ou une autre valeur si elle est fausse.
- Sa syntaxe est la suivante :

```
$var = expression ? valeur1 : valeur2
```

Elle est équivalente à :

```
if(expression) {$var=valeur1;}
else {$var=valeur2;}
```

L'opérateur ?

- Le premier exemple de calcul de remise pourrait s'écrire :

```
$var = ($prix>100)? "la remise est de 10 %":"la remise est de 5 %";
echo "<b>Pour un montant d'achat de $prix &euro;: $var </b><br />";
```

- au lieu de :

```
if($prix>100)
{
  echo "<b>Pour un montant d'achat de $prix &euro;, la remise est de 10 %</b><br />";
}
else
{
  echo "<b>Pour un montant d'achat de $prix &euro;, la remise est de 5 %</b><br />";
}
```

Exemple 5-3. L'opérateur ?

```php
<?php
$ch = "Bonjour ";
$sexe="M";
$ch .= ($sexe=="F")?"Madame":"Monsieur";
echo "<h2>$ch</h2>";
$nb = 3;
$pmu ="Il faut trouver ".$nb;
$mot = ($nb==1)?" cheval":" chevaux";
echo "<h3> $pmu $mot </h3>";
?>
```

Compte tenu des valeur des variables `$sexe` et `$nb` le résultat retourné est le suivant :

```
Bonjour Monsieur
Il faut trouver 3 chevaux
```

Exemple

- <?php
- $age = 8;
- if ($age <= 12) // SI l'âge est inférieur ou égal à 12
- { echo "Salut ! Bienvenue sur mon site !
";
- $autorisation_entrer = "Oui";}
- else // SINON
- { echo "Ceci est un site pour enfants, vous êtes trop vieux pour pouvoir entrer. Au revoir !
";
- $autorisation_entrer = "Non";}
- echo "Avez-vous l'autorisation d'entrer ? La réponse est : $autorisation_entrer";
- ?>

Exercice(Opérateur ternaire)

```
<?php
$age = 24;
if ($age >= 18)
{
    $majeur = true;
}
else
{
    $majeur = false;
}
?>
```

```
<?php
$age = 24;

$majeur = ($age >= 18) ? true : false;
?>
```

Exemple'AND'

- <?php
- if ($age <= 12 AND $sexe == "garçon")
- {
- echo "Bienvenue sur le site de Captain America!";
- }
- elseif ($age <= 12 AND $sexe == "fille")
- {
- echo "C'est pas un site pour les filles ici, retourne jouer à la Barbie !";}
- ?>

Exemple'OR'

```
<?php
if ($sexe == "fille" OR $sexe == "garçon")
{
    echo "Salut Terrien !";
}
else{
    echo "Euh, si t'es ni une fille ni un garçon, t'es quoi alors ?";
}
?>
```

L'instruction switch...case

- Supposez que vous vouliez associer un code de département avec son nom réel. Avec une suite d'instructions if, vous écririez le script suivant :
  - <?php
  - $dept=48;
  - if($dept==48) echo "Oujda";
  - if($dept==50) echo " Meknes";
  - if($dept==1) echo " Rabat";
  - if($dept==20) echo "Fes";
  - ?>

L'instruction switch...case

- dans lequel la variable $dept proviendrait d'un formulaire, par exemple.
- Ce code peut être simplifié sans multiplier les instructions if grâce à l'instruction switch...case. Cette dernière permet de comparer la valeur d'une expression avec une liste de valeurs prédéterminées par le programmeur et d'orienter le script en fonction de la valeur de cette expression.

Syntaxe de base : Les instructions conditionnelles(2)

- L'instruction switch

```
switch (Variable)
 {
case Valeur1:
// Liste d'instructions 1
 break;
case Valeur2: Liste d'instructions break;
...
case Valeurs N:
// Bloc d'instructions N
 break;
default:
//  Liste d'instructions par defaut
 break;  }
```

Explication

- Si l'expression qui suit le mot-clé switch vaut valeur1, les instructions qui suivent la première instruction case sont exécutées, après quoi l'exécution passe à la fin du bloc switch. Il en va de même pour les valeurs suivantes. Si aucune concordance n'est trouvée, ce sont les instructions qui suivent l'instruction default qui sont exécutées.

Exemple

```
<?php
$note = 10;
switch ($note) // on indique sur quelle variable on travaille
{ case 0: // dans le cas où $note vaut 0
echo "Tu es vraiment un gros Zér0 !!! " ;break;
case 5: // dans le cas où $note vaut 5
echo "Tu es très mauvais " ;break;
case 7: // dans le cas où $note vaut 7
echo "Tu es mauvais " ;break;
case 10: // etc etc
echo "Tu as pile poil la moyenne, c'est un peu juste... " ;break;
case 12:echo "Tu es assez bon " ;break;
case 16:echo "Tu te débrouilles très bien ! " ;break;
case 20:echo "Excellent travail, c'est parfait ! " ;break;
Default:echo "Désolé, je n'ai pas de message à afficher pour cette note "
;}
?>
```

Remarque

- Plusieurs instructions case différentes peuvent se succéder avant qu'intervienne un bloc d'instructions (voir repère 1 dans l'exemple 5-4). Dans ce cas, les différentes valeurs indiquées déclenchent l'exécution du même code (repère 2).

Exemple 5-4. L'instruction switch…case

```
<?php
$dept=1;
switch($dept)
{ //Premier cas
Case 1:                 → 1
case "Capitale":  →1
echo " Rabat";    →2
break;
case 48:
case " Oriental":
echo "Oujda";
break;
//la suite des départements...
//Cas par défaut
default:
echo "Département inconnu en Ile de Maroc";
break; }    ?>
```

Les instructions de boucle

- Les boucles permettent de répéter des opérations élémentaires un grand nombre de fois sans avoir à réécrire le même code. Selon l'instruction de boucle utilisée, le nombre d'itérations peut être défini à l'avance ou être déterminé par une condition particulière.

Syntaxe de base : Les instructions de boucle

- La boucle for
  - for(expression1; expression2; expression3)
  - { //instruction ou bloc; }
- La boucle while
  - While(condition)
  - {bloc d'instructions ;}
- La boucle do…while
  - Do {bloc d'instructions ;}while(condition) ;
- La boucle foreach (PHP4)
  - Foreach ($tableau as $valeur)
  - {insts utilisant $valeur ;}

La boucle for

- Présente dans de nombreux langages, la boucle for permet d'exécuter plusieurs fois la même instruction ou le même bloc sans avoir à réécrire les mêmes instructions.

La boucle for

- expression1 est toujours évaluée. Il s'agit généralement de l'initialisation d'une ou plusieurs variables servant de compteur pour la boucle. expression2 est ensuite évaluée avec une valeur booléenne : si elle vaut TRUE, la boucle continue et les instructions comprises dans le bloc sont exécutées, sinon la boucle s'arrête. Si elle est toujours vraie on obtient une boucle infinie, vérifiez donc qu'elle peut être fausse. expression3 n'est exécutée qu'à la fin de chaque itération. Il s'agit le plus souvent d'une instruction d'incrémentation de la variable compteur.

Exemples

```php
<?php
/* exemple 1 */

for ($i = 1; $i <= 10; $i++) {
    echo $i;
}

/* exemple 2 */

for ($i = 1; ; $i++) {
    if ($i > 10) {
        break;
    }
    echo $i;
}
```

```php
/* exemple 3 */

$i = 1;
for (; ; ) {
    if ($i > 10) {
        break;
    }
    echo $i;
    $i++;
}

/* exemple 4 */

for ($i = 1, $j = 0; $i <= 10; $j += $i, print $i, $i++);
?>
```

Exemple(for)

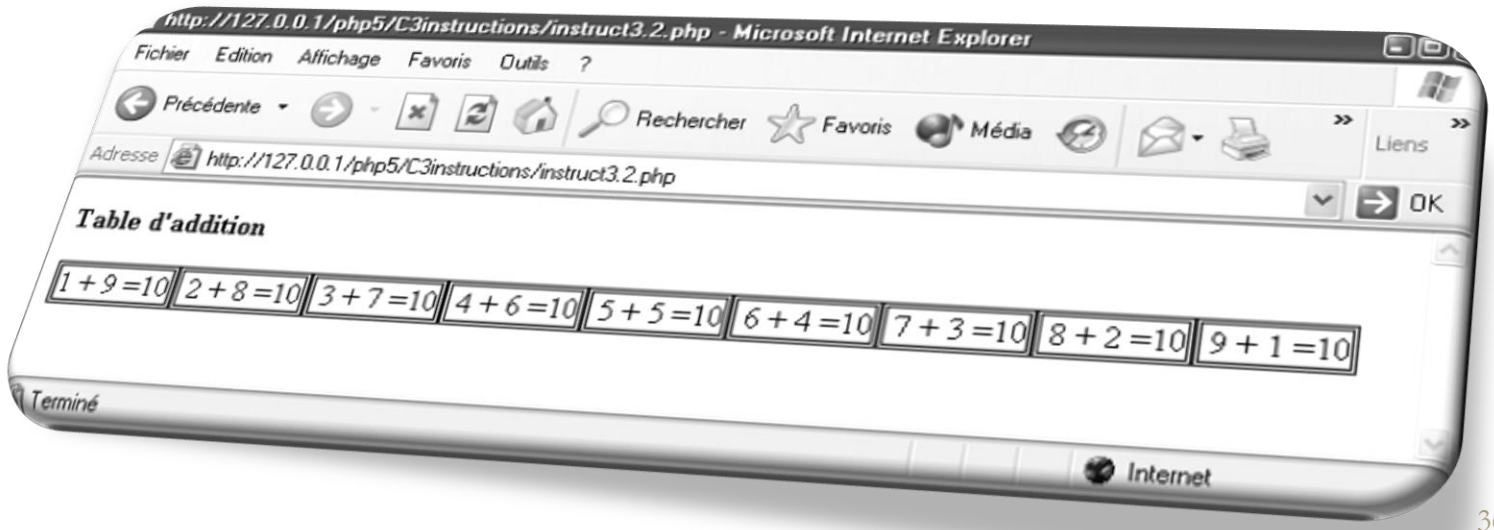
- <?php
- for ($nombre_de_lignes = 1; $nombre_de_lignes <= 10; $nombre_de_lignes++)
- {
- echo 'Ceci est la ligne n°' . $nombre_de_lignes . '
';
- }
- ?>

Une boucle à plusieurs variables.

- Les trois expressions utilisées dans la boucle for peuvent contenir plusieurs parties séparées par des virgules. La boucle peut en ce cas être réalisée sur plusieurs variables, comme illustré à l'exemple précédent.

Exemple 5-6

```php
<?php
for($i=1,$j=9;$i<10,$j>0;$i++,$j--)
//$i varie de 1 à 9 et $j de 9 à 1
{
echo "<span style=\"border-style:double;border-width:3;\"> $i + $j=10</span>";
}
?>
```



Les boucles imbriquées

- Il est possible d'imbriquer des boucles for les unes dans les autres sur autant de niveaux que désiré, le bloc qui suit la première contenant toutes les autres. Chaque variable compteur déclarée dans une boucle n'est utilisable que dans la boucle qui la déclare et dans celles de niveau inférieur.

La boucle while

- La boucle for oblige à préciser les valeurs limites pour lesquelles la boucle va s'arrêter.
- À moins d'utiliser une instruction if pour la stopper à l'aide de l'instruction . il faut connaître ces valeurs limites. La boucle while permet d'affiner ce comportement en réalisant une action de manière répétitive tant qu'une condition est vérifiée ou qu'une expression quelconque est évaluée à TRUE et donc de l'arrêter quand elle n'est plus vérifiée (évaluée à FALSE).

Exemple(while)

```
<?php
$nombre_de_lignes = 1;
while ($nombre_de_lignes <= 100)
{
    echo 'Je ne dois pas regarder les mouches voler quand j\'apprends le PHP.<br />';
    $nombre_de_lignes++;
// $nombre_de_lignes = $nombre_de_lignes + 1
}
?>
```

Exemple

```
<?php
$nombre_de_lignes = 1;
while ($nombre_de_lignes <= 10)
{
echo 'Ceci est la ligne n°' .
$nombre_de_lignes . '<br />';
$nombre_de_lignes++;
}
?>
```

La boucle do...while

- La boucle do...while apporte une précision à la boucle while. Dans celle-ci, en effet, si l'expression booléenne est évaluée à FALSE, les instructions qu'elle contient ne sont jamais exécutées. Avec l'instruction do...while, au contraire, la condition n'est évaluée qu'après une première exécution des instructions du bloc compris entre do et while.

Exemple

- <?php
- $nombre_de_lignes = 1;
- Do {
- echo 'Ceci est la ligne n°' . $nombre_de_lignes . '
';
- $nombre_de_lignes++;
- } while ($nombre_de_lignes <= 10)

- ?>

La boucle foreach

- Introduite à partir de la version 4.0 de PHP, l'instruction foreach permet de parcourir rapidement l'ensemble des éléments d'un tableau, ce que fait aussi une boucle for, mais foreach se révèle beaucoup plus efficace.
- La boucle foreach est particulièrement efficace pour lister les tableaux associatifs dont il n'est nécessaire de connaître ni le nombre d'éléments ni les clés. Sa syntaxe est variable selon que vous souhaitez récupérer seulement les valeurs ou les valeurs et les clés (ou les indices).
- Pour lire les valeurs seules, la première syntaxe est la suivante :

```
foreach($tableau as $valeur)
{
//bloc utilisant la valeur de l'élément courant
}
```

La boucle foreach

- La variable $valeur contient successivement chacune des valeurs du tableau. Il importe cependant de ne pas utiliser un nom de variable existant, faute de quoi sa valeur est écrasée.
- Les variables utilisées dans une boucle foreach ne sont pas locales à la boucle et gardent donc la valeur du dernier élément lu dans tout le script.
- Pour lire les valeurs et les clés (ou les indices), la deuxième syntaxe est la suivante :

```
foreach($tableau as $cle=>$valeur)
{
//bloc utilisant la valeur et la clé de l'élément courant
}
```

Exemple 5-7. Lecture des valeurs d'un tableau indicé

```php
<?php
//Création du tableau de 9 éléments
for($i=0;$i<=8;$i++)
{
  $tab[$i] = pow(2,$i); ←1
}
$val ="Une valeur";
echo $val,"<br />";
//Lecture des valeurs du tableau
echo"Les puissances de 2 sont :";
foreach($tab as $val) ←2
{echo $val." : ";}
?>
```

Le résultat suivant est affiché :

```
Les puissances de 2 sont :1 : 2 : 4 : 8 : 16 : 32 : 64 : 128 : 256:
```

Exemple 5-8. Lecture des indices et des valeurs

```php
<?php
//Création du tableau
for($i=0;$i<=8;$i++)
{
  $tab[$i] = pow(2,$i);
}
//Lecture des indices et des valeurs
foreach($tab as $ind=>$val) ←❶
{echo " 2 puissance $ind vaut $val <br />";}
echo "Dernier indice ",$ind, " ,dernière valeur ",$val; ←❷
?>
```

```
2 puissance 0 vaut 1
2 puissance 1 vaut 2
.....................
2 puissance 7 vaut 128
2 puissance 8 vaut 256
Dernier indice 8 ,dernière valeur 256
```

Sortie anticipée des boucles

- Vous pouvez avoir besoin d'arrêter une boucle avant son terme normal. Pour cela, vous disposez des instructions break et continue, qui permettent de réaliser un arrêt partiel ou total.
- L'instruction break
- L'instruction continue

Exercices

- For et while:
  - Choisir un nombre de trois chiffres. Effectuer ensuite des tirages
    aléatoires et compter le nombre de tirages nécessaire pour obtenir
    le nombre initial. Arrêter les tirages et afficher le nombre de coups
    réalisés. Réaliser ce script d'abord avec l'instruction while puis
    avec l'instruction for.

Exercices

- For et while:
  - Choisir un nombre de trois chiffres. Effectuer ensuite des tirages aléatoires et compter le nombre de tirages nécessaire pour obtenir le nombre initial. Arrêter les tirages et afficher le nombre de coups réalisés.
  - Réaliser ce script d'abord avec l'instruction **while** puis avec l'instruction **for**.

FIN

- Merci pour votre attention