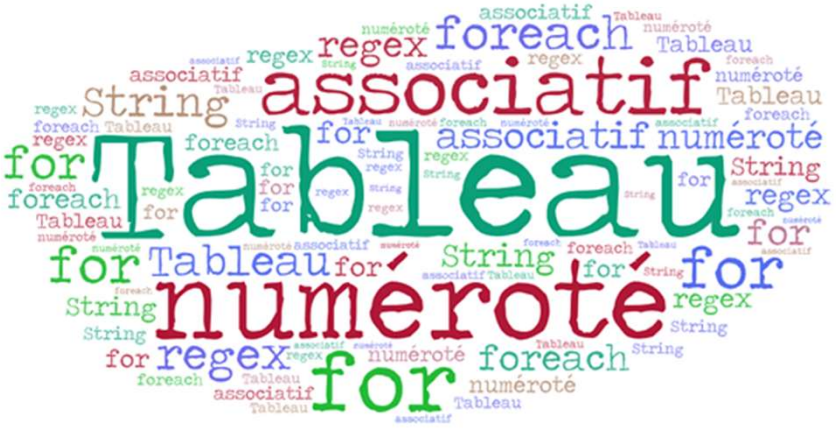
PROGRAMMATION WEB

PROGRAMMATION WEB

Cours 2:

Les tableaux en PHP



Mansour Sihem

LA DÉCLARATION DES TABLEAUX EN PHP

- La déclaration d'un tableau se fait de la même manière qu'une variable, c'est à dire avec un signe dollars ($) et un nom.
- En PHP, on trouve deux types de tableaux numéroté et associatif

LES TABLEAUX NUMÉROTÉS:

- Pour créer un tableau numéroté, on utilise la syntaxe suivante

```php
$tab=[1,2,3,4,5];
echo $tab[0];
```

- Un tableau peut avoir un contenu de plusieurs types, voir l'exemple:

```php
<?php
    $tab=[1,'essai',true,4.5,5];
    $i;
    for($i=0;$i<5;$i++) echo $tab[$i].'<br>';
?>
```

PROGRAMMATION WEB

LES TABLEAUX NUMÉROTÉS:

- On peut aussi créer le tableau manuellement case par case
- Voir l'exemple ci-dessous:

```
$matiere[0] = 'PHP';
$matiere[1] = 'Java';
$matiere[2] = 'stats';
```

- Ou:

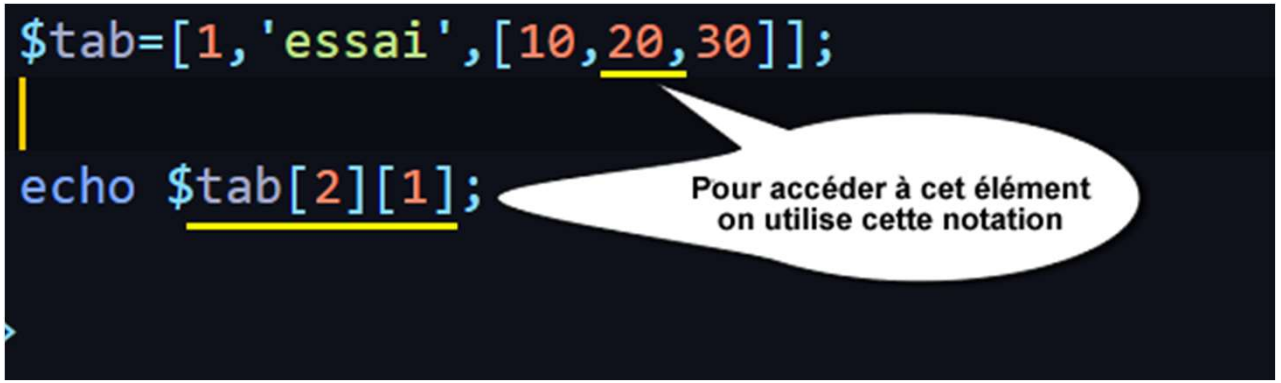
```
$matiere[] = 'PHP';
$matiere[] = 'Java';
$matiere[] = 'stats';
```

- L'indexation du tableau numéroté commence toujours par 0, vous n'êtes obligés de suivre un ordre précis pour remplir un tableau numéroté, voir l'exemple ci-dessous

LES TABLEAUX NUMÉROTÉS:

```php
$matiere[0] = 'PHP';
$matiere[1] = 'Java';
$matiere[2] = 'stats';
$matiere[10] = 'angl';
```

- on peut avoir une ou plusieurs cases tableaux dans le tableau lui-même, voir l'exemple suivant pour voir comment accéder à ses éléments:



LES TABLEAUX ASSOCIATIFS:

- Les tableaux associatifs se comportent comme les tableaux numérotés la seule différence est que ses cases sont indexées par des noms au lieu des valeurs numériques
- Voir l'exemple ci-dessous:

```php
$elv=['nom'=>'essai',
      'mat'=>1,
      'note'=>[10,8,15]
     ];
```

- On peut créer les tableau associatif comme l'exemple suivant

```php
$coordonnees['prenom'] = 'Ali';
$coordonnees['nom'] = 'dridi';
$coordonnees['adresse'] = '3 Rue du Paradis';
$coordonnees['ville'] = 'Tunis';
```

- Pour accéder à des éléments de ce tableau, voir l'exemple suivant:

```php
echo $elv['nom'].'<br>';
echo $elv['note'][2];
```

LES TABLEAUX ASSOCIATIFS:

- La déclaration du tableau associatif peut prendre cette forme:

```
$etu=[
    [
    'mat'=>1,
    'nom'=>'flen',
    'note'=>[9,16,12,6]
    ],
    [
        'mat'=>2,
        'nom'=>'flena',
        'note'=>[5,16,12,14]
    ],
    [
        'mat'=>3,
        'nom'=>'yy',
        'note'=>[2,15,12,7]
        ]
];
```

PARCOURS DES TABLEAUX.

- Pour parcourir un tableau, on peut utiliser trois manières différentes:
  - la boucle for ;
  - la boucle foreach ;
  - On peut utiliser la fonction built-in **print_r** qui permet d'afficher le tableau suivant un format fixé par le langage

PARCOURS DES TABLEAUX.

- La boucle for: il est utilisé facilement avec les tableau numéroté.
- Voir l'exemple ci-dessous:

```
$i;
$tab=[1,4,'essai',2.5];
for($i=0;$i<count($tab);$i++)echo $tab[$i].'</br>'
```

- Cette boucle n'est pas adaptée aux tableaux associatifs,

PARCOURS DES TABLEAUX.

- La boucle foreach: cette boucle est adaptée à tous les types de tableaux, facile et ne nécessite pas une fonction qui détermine la taille du tableau,
- Voir l'exemple ci-dessous pour voir comment appliquer cette boucle sur un tableau numéroté:

```php
$tab=[1,true,'essai',2.5];
foreach($tab as $val) echo $val.'<br>';
```

PARCOURS DES TABLEAUX.

- Voir l'exemple ci-dessous pour voir comment appliquer la boucle foreach sur des tableaux associatifs

```php
$elv=['nom'=>'essai',
        'mat'=>1,
        'note'=>10
      ];
  foreach($elv as $key => $val) echo $key.' => '.$val.'<br>';
```

- Voir le deuxième exemple, la fonction built-in **gettype()** prend en paramètre une variable et retourne son type,

```php
$elv=['nom'=>'essai',
        'mat'=>1,
        'note'=>[10,12,14]
      ];
  foreach($elv as $key => $val) {
      if(gettype($val)!='array') echo $key.' => '.$val .'<br>';
      else {
          echo $key.'<br>';
          foreach($val as $v) echo $v.'<br>';
      }
  }
```

PARCOURS DES TABLEAUX.

- Voir un autre exemple:

```php
$elv=[
        [   'mat'=>1,
            'nom' => 'essai',
            'notes'=>[10,12,14]

        ],
        [   'mat'=>2,
            'nom'=> 'flen',
            'notes'=>[10,11,15]

        ]
    ];
```

- Voir comment afficher son contenu

```php
foreach ($elv as $key ) {
    foreach($key as $t => $v)
    if(gettype($v)!='array') echo $v.'<br>';
    else {
        echo $t.'<br>';
        foreach($v as $e)echo $e.'<br>';

    }
```

PARCOURS DES TABLEAUX.

- Afficher rapidement un array avec print_r: cette instruction est une sorte d'echo spécialisé pour les arrays.
- Voir l'exemple:

```php
$coordonnees['prenom'] = 'Ali';
$coordonnees['nom'] = 'dridi';
$coordonnees['adresse'] = '3 Rue du Paradis';
$coordonnees['ville'] = 'Tunis';

echo '<pre>';

print_r($coordonnees);

echo '</pre>';
```

Resultat

```
Array
(
    [prenom] => Ali
    [nom] => dridi
    [adresse] => 3 Rue du Paradis
    [ville] => Tunis
)
```

QUELQUES FONCTIONS UTILES POUR LES TABLEAUX.

- str_split: transforme une chaine de caractères à un tableau de caractères
- count() et sizeof() retournent toutes les deux la taille du tableau passé en paramètre.
- sort()/ rsort() trie les éléments d'un tableau, la première fait le tri croissant la deuxième fait le tri décroissant
- array_key_exists : pour vérifier si une clé existe dans le tableau ;
- in_array : pour vérifier si une valeur existe dans l'array ;
- array_search : pour récupérer la clé d'une valeur dans l'array. Exemple:

```
$position = array_search('Fraise', $fruits);
echo '"Fraise" se trouve en position ' . $position . '<br />';
```

LES CHAINES DE CARACTÈRES.

- Une chaine de caractères constante est délimitée par des apostrophes ou des guillemets

```php
<?php
    $str='hello';
    $str1="Essai";
    echo $str;
    echo $str1;

?>
```

- La concaténation entre les chaines se fait en utilisant le symbole . Voir l'exemple suivant:

```php
<?php
    $str='hello';
    $str1="Essai";
    echo 'Voici le contenu de deux chaines '.$str.' '.$str1;
?>
```

LES CHAINES DE CARACTÈRES.

- Pour accéder à un caractère bien défini dans une chaine de caractère, voir l'exemple suivant:

```php
<?php
    $ch="essai";
   echo $ch[1];


?>
```

- Voici un deuxième exemple;

```php
<?php
    $ch="essai";
    for($i=0;$i<strlen($ch);$i++) echo$ch[$i].' ';
?>
```

QUELQUE FONCTIONS UTILES SUR LES CHAINES DE CARACTÈRES

- **strlen($str)**: retourne un entier qui représente le nombre de caractères que contient la chaîne $str.
- **strtoupper($str)**: convertir la chaîne $str en majuscule.
- **strtolower($str)**: convertir la chaîne $str en minuscule.
- **ucfirst($str)**: convertit le premier caractère de la chaîne $str en majuscules.
- **trim($str)**: supprime les espaces de début de de fin de la chaîne $str.
- **rtrim($str)** ou **chop($str)**: supprime les espaces de fin de la chaîne $str.
- **substr($str,$deb,$nbr)**: extrait une partie de la chaîne de caractères en commençant de la position $deb et en retournant $nbr caractères (Notez que la position du premier caractère de la chaîne est 0).

LES EXPRESSIONS RÉGULIÈRES: REGEX

Les expressions régulières, ou plus communément regex (contraction de regular expression) permettent de représenter des modèles de chaînes de caractère. En php, on a deux familles de regex : POSIX est l'acronyme de Portable Operating System Interface et PCRE signifie Perl Compatible Regular Expressions.

LES EXPRESSIONS RÉGULIÈRES: REGEX

Dans notre cours, nous allons utilisé la famille regex PCRE et essentiellement la fonctions built-in: preg_match(pattern, string) qui prend deux paramètres obligatoires : l'expression régulière et la chaine de caractères. La fonction retourne 1 si le résultat est vrai sinon elle retourne 0.

```php
<?php
    $str='hello';
    if(preg_match("/(hi)/",$str)) echo '<p style="color:green;"> ok </p>';
    else echo '<p style="color:red;"> non </p>';

?>
```

Vous pouvez utiliser ce site pour faire le test de vos expressions régulières: https://regex101.com/

LES EXPRESSIONS RÉGULIÈRES: REGEX

- Voici la liste des symboles qu'on peut utiliser dans la composition des expressions régulières

OPÉRATEUR	DESCRIPTION
^	Il indique le début d'une chaîne de caractères.
$	Il indique la fin d'une chaîne.
.	Il indique presque n'importe quel caractère.
()	Il indique un groupe d'expressions.
[]	Il trouve une plage de caractères, par exemple [xyz] signifie x, y ou z .
[^]	Il trouve les éléments qui ne sont pas dans la plage, par exemple [^abc] signifie tout sauf a, b ou c.
- (tiret)	Trouve la plage de caractères dans la plage d'éléments donnée, par exemple [a-z] signifie de a à z.
| (pipe)	C'est le OU logique par exemple x | y signifie x OU y.
?	Il indique zéro ou un caractère précédent.
*	Il indique zéro ou plusieurs caractères précédents.
+	Il indique un ou plusieurs caractères précédents.
{n}	Il indique exactement n fois la plage de caractères précédents.
{n, }	Il indique au moins n fois le caractère ou la plage d'éléments précédents.
{n, m}	Il indique au moins n mais pas plus de m fois, par exemple x{2, 4} signifie x répété 2 à 4 fois.
\	Il indique le caractère d'échappement.

LES EXPRESSIONS RÉGULIÈRES: REGEX

- Voici la liste des caractères spéciaux qu'on peut utiliser dans la composition des expressions régulières

CARACTÈRE SPÉCIAL	DESCRIPTION
\n	Il indique une nouvelle ligne.
\r	Il indique un retour de chariot.
\t	Il indique une tabulation.
\s	Correspond aux caractères d'espacement comme l'espace, le saut de ligne ou la tabulation.
\d	Correspond à n'importe quel chiffre de 0 à 9.
\w	Correspond à des les lettres minuscules, majuscules, les chiffres et le caractères de soulignement.

- Source: https://waytolearnx.com/2020/01/les-expressions-regulieres-en-php.html

LES EXPRESSIONS RÉGULIÈRES: REGEX

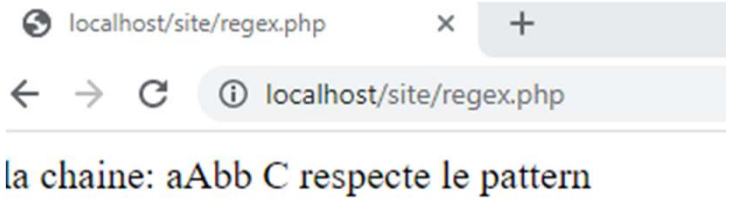
- Exemple d'utilisation:

EXPRESSION RÉGULIÈRE	CORRESPONDANCE
xxx	La chaîne « xxx »
^xxx	La chaîne qui commence par « xxx »
xxx$	La chaîne qui se termine par « xxx ».
^xxx$	La chaîne ne contient que « xxx ».
he*llo	Correspond à llo, hello ou hehello, mais pas hellooo
hello+	Il correspondra à hello ou plus. Par exemple, hello, hellohello ou hellohellohello
he?llo	Il correspondra à llo ou hello
he.o	Il correspondra à n'importe quel caractère entre he et o. Les correspondances possibles sont helo ou hero, mais pas hello
[abc]	a, b, ou c
[a-z]	Correspond à n'importe quelle lettre minuscule
[^A-Z]	Toute les lettres sauf les lettres majuscules
(a|b)	Soit « a » ou « b »
[a-z]+	Une ou plusieurs lettres minuscules
^[a-zA-Z0-9._-]	Correspond à n'importe quelle lettre minuscule ou majuscule, à des chiffres entre 0 et 9 et à des points, des caractères de soulignements ou des tirets.
+@[a-zA-Z0-9-]	Correspond au symbole @ suivi de lettres minuscules ou majuscules, des chiffres entre 0 et 9 ou des tirets.
+\.[a-zA-Z.]{2,5}$/	Échappe le point à l'aide de \, puis fait correspondre toute lettre minuscule ou majuscule ayant une longueur de caractère comprise entre 2 et 5 à la fin de la chaîne.

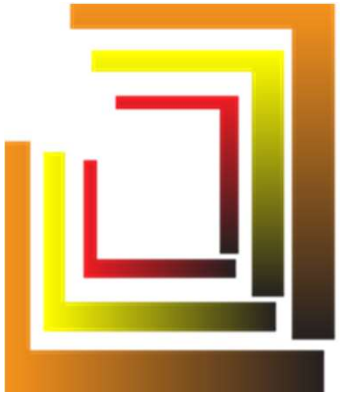
LES EXPRESSIONS RÉGULIÈRES: REGEX

- Exemple d'utilisation: dans cet exemple la chaine fournie doit commencer par un caractère alphabétique puis un ensemble de caractères alphabétiques et espace,
- Avec l'option i, la casse n'est pas respectée (la chaine peut être écrite en minuscule comme en majuscule)

```php
<?php
    $pattern='/^[a-z][a-z\s]*$/i';
    $str='aAbb C';
    if(preg_match($pattern,$str)) echo 'la chaine:  '.$str.' respecte le pattern';
    else echo 'la chaine:  '.$str.'ne respecte pas le pattern';
?>
```



PROGRAMMATION WEB



FIJN

Mansour Sihem