20/06/2023

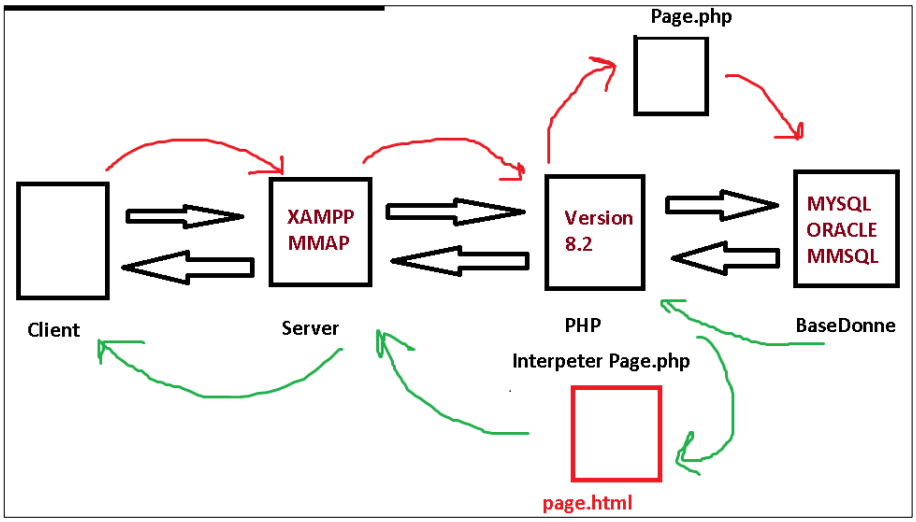
PHP

Basic et Mysql

youssef khanchoufi

1) Introduction

1. Dernier Version Php : 8.2.7 (2023)



2) Installation

2. Serveur Web

Web-Server : **Apache** et **nginx** et **IIS**

- Nginx : Prémier : **2004** () / Dernier_version (1.25.1 = 2023) / open_source / server logiciel
- Apache : Prémier : **1995** / Dernier_version (10.0 = 2023) / open_source / server logiciel
- ISS : Microsoft / Dernier_version (2.4.47 = 2015)

WAMP (windows) (Apache Http server , Mysql , MariaDB , Php , perl)

XAMPP (windows, macOS et linux) (Apache Http server , Mysql , MariaDB , Php , perl, phpmyadmin, FTP, FileZilla)

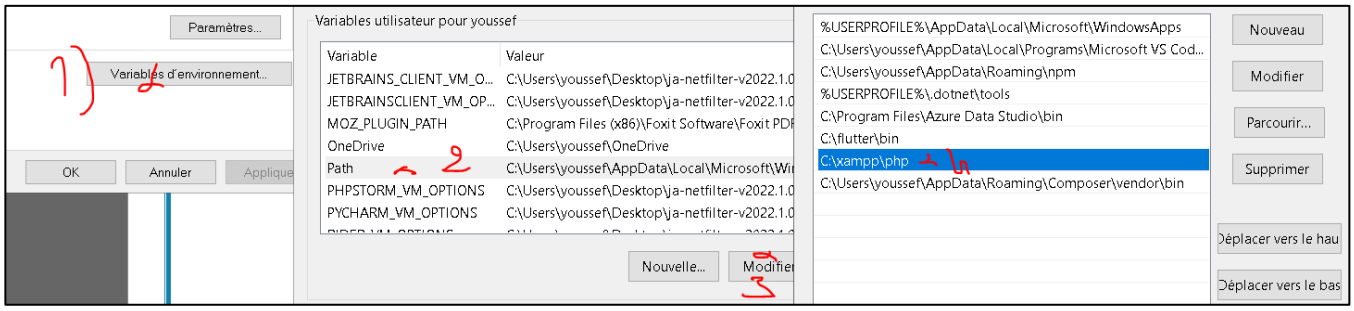
3. Installation

Telecharger **php** dans site web : https://www.php.net/downloads

Taper : modifier les variable d'environnement dans recherche

Click sur : variables d'environment.

Modifier variable path . Ajouter nouvelle **path** vers dossier **php 8**.



```
C:\Users\youssef>php -v
PHP 8.1.2 (cli) (built: Jan 19 2022 10:18:23) (ZTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.1.2, Copyright (c) Zend Technologies

C:\Users\youssef>echo %path% Verifier les paths qui exists
C:\Program Files\Python311\Scripts\;C:\Program Files\Python311\;C:\Program Files\Microsoft MPI\Bin\;C:\WINDOWS\system32;
C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;C:\WINDOWS\System32\OpenSSH\;C:\Program
Files\Microsoft SQL Server\130\Tools\Binn\;C:\Program Files\Git\cmd;C:\xampp\php;C:\Program Files\nodejs\;C:\Program Fil
es\dotnet\;C:\Program Files (x86)\Microsoft SQL Server\130\Tools\Binn\;C:\Program Files (x86)\Microsoft SQL Server\130\D
TS\Binn\;C:\Program Files\Microsoft SQL Server\130\DTS\Binn\;C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\T
ools\Binn\;C:\Program Files (x86)\Microsoft SQL Server\150\DTS\Binn\;C:\Program Files\Azure Data Studio\bin;C:\ProgramDa
ta\ComposerSetup\bin;C:\Windows\System32;C:\Users\youssef\AppData\Roaming\Composer\vendor\bin;C:\Windows\System32;C:\Use
rs\youssef\AppData\Local\Programs\Microsoft VS Code\bin;C:\Users\YourUsername\AppData\Roaming\Composer\vendor\bin;

C:\Users\youssef>set PATH=%PATH%;C:\xampp\php Ajouter Path si exit un problem
```

Dans CMD Vérifier Si installer :

4. PHP dans Live server Vs-Code :

- https://www.youtube.com/watch?v=joxLj3O_QGo = Php dans Live server VS code

3) Syntaxe

- Php traireter le code entre balise <?php ?>
- Balise courte <?= $a ?-> == <?php echo "" ?->
- Les instructions PHP se terminent par un point-virgule (;)
- Si une ligne dans <?php ?> pas besion point-virgule <?php echo "Youssef" ?>
- Si un fichier contient seulement du code PHP, Balise de fermuture ?->

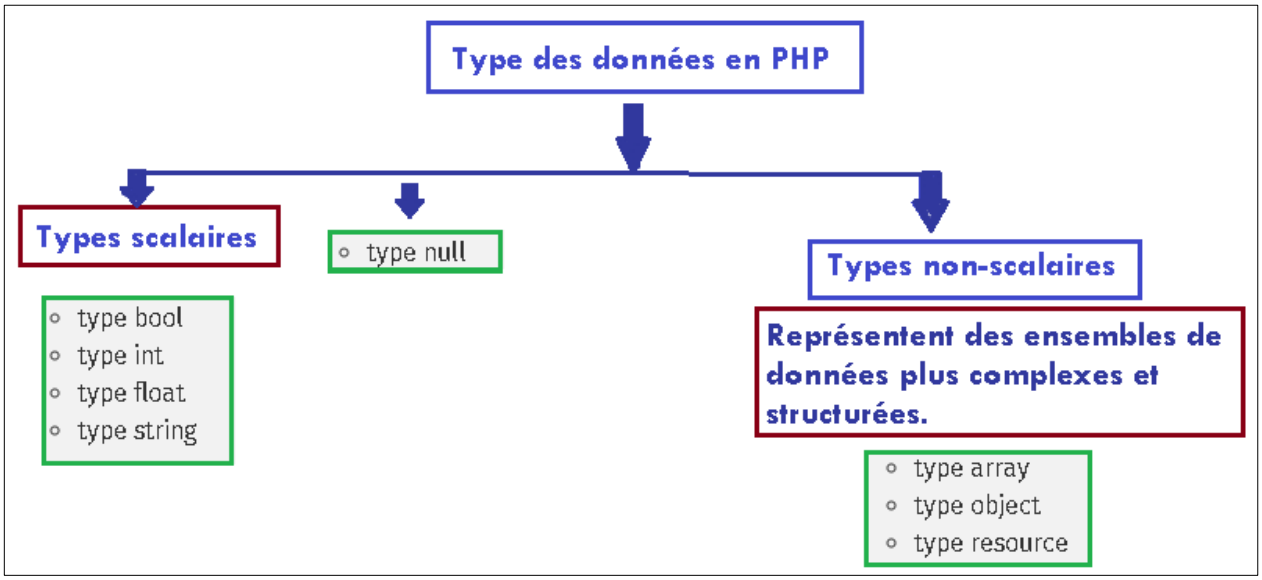
4) Echo / Print : Pour afficher les données

```
Echo :
------
- Deux syntaxe : echo("Expression") ou echo "expression"
- plusieurs arguments séparés par des virgules : echo "ch1" , "ch2" , "ch3"
- N'a pas de valeur de retour, elle renvoie toujours void (rien).
- Plus Rapide

Print :
-------
- Deux syntaxe : print("expression") ou print "expression"
- Renvoi 1 en cas de succès ou 0 en cas d'échec.
```

5) Variables :

5. Type des données en PHP :




```
Type Boolean :
-------------
- Booelan littéal  : deux valeurs : true / false.
- Résultat d'un opérateur retourne un booléen. Suite Passé à une Structure de Controle.
- Quand utilise dans contexte logique, elle sera automatiquement interpétée comme une valeur
de type bool.  $bool = "";  if($bool){ echo "success"; } donc $bool = false.
- Conversion : (false , 0 , 0.0 , -0.0 , "vide" , "0" , array[] , NULL ) = False
- les autres type est   : true.
- echo false  => chaine : "vide".
- echo true   => chaine :  "1" .
- is_bool(variable)     : Vérifie si une valeur est de type booléen.
```

```
Type Float :
------------
- Les nombres décimaux (nombres à virgule flottante ) positifs ou négatifs.

Formes pour représenter des valeurs à virgule flottante :
---------------------------------------------------------

- Notation normale      : Point décimal pour séparer les deux partie [18.5]
- Notation scientifique : exposant (e) puissance de 10  [13e3 = 13000] [12e-2 = 0.12]
- Conversion de chaine en nombre flottant :  floatval("12.5")  , (float) "15.6"

* Le fonctions :

- Ceil   (num)  : Retourne l'entier supérieur la plus proche ou égal.
- Floor  (num)  : Retourne l'entier inférieur la plus proche ou égal.
- Round(num, précision)    : Retourne la valeur arrondie le plus proche.  à la précision precision.
           * Si la Partie decimal est inférieure à 0.5          :  arrondi à la valeur inférieure.
           * Si la partie décimale est supérieure ou égale à 0.5 :  arrondi à la valeur supérieure.
           * précision :
                      - Positive : Digits après le point décimal.
                      - negative : Digits avant le point décimal.

- Type de retour ( Ceil , Floor , Round ) : est (float)

- Sprintf    : printf("%.2f", nombre)
- Number_format(nombre , le nombre de valeurs décimales après le point décimal)

- Conversion :
 * Si une chaîne est - numérique - ou - numérique de tête -
   -alors elle sera transformée en sa valeur flottante correspondante.

Exmp : ["12.5"] = 12.5      ,   ["20.8sfsdf"] = 20.8
```

```
Type Entier :
-------------
- Un entier est un nombre appartenant à l'ensemble Z= {-2,-1,0,2,5,..... }.
- Octal        : 0o....
- Hexadecimal  : 0x....
- Binaire      : 0b....
- PHP_INT_MAX , PHP_INT_MIN , PHP_INT_SIZE  = 8 byte =
- Syntaxe      : 2000000 => 2_000_000. var_dump(2_000_000) = 2000000.
- Valeur limite PHP_INT_Size stockée en tant que float.
```

```
Chaine De Caracatére :
----------------------
- Une chaîne de caractères littérale peut être spécifiée de 4 façons différentes :

1) Guillemets simples :
        - Les chaînes de caractères littérales peuvent être définies entre guillemets simples '' .
        - Les caractères spéciaux, l'apostrophe (').    \'  =  '
        - Les noms de variables ne seront pas interprétés

2) Guillemets doubles :
        - Les chaînes de caractères littérales peuvent être définies entre guillemets doubles "" .
        - "Hamza" . Séquences d'échappement [ \n \t \\ \" \$ \t \ ]
        - Analyse les variables : Les noms de variables seront interprétés

3) Syntaxe Heredoc  :
        - Définir des blocs de texte sans avoir besoin d'échapper les caractères spéciaux ou d'interpoler des Var.
        - La syntaxe est <<<IDENTIFICATEUR   Texte   IDENTIFICATEUR;
        - Régle   : L'identificateur de fermeture ne doit pas être en retrait plus loin que les lignes du corps
        - Analyse les variables : Les noms de variables seront interprétés
        - Régle du Nom comme des variable : peut contenir [A-Z] [a-z] [0-9] commencer par Char non Numérique ou [_]
        - selement les séquences d'échappement ; \n \t \r \$ \\

4) Syntaxe Nowdoc  :
        - La syntaxe est <<<'IDENTIFICATEUR'  \nTexte\n  IDENTIFICATEUR;.
        - Régle   : L'identificateur de fermeture ne doit pas être en retrait plus loin que les lignes du corps

- Analyse des variable :
-----------------------
- Syntaxe Simple   : $var = 5; "Hamza $varIlyass"; mais errore pour connaitre nom de variable $var.
- Syntaxe Complexe : l'utilisation d'accolades autour de l'expression. "Je suis {$var}Ilyas"; Valide.
```

```
Type NULL :
-----------
- Les variables non définies et les variables unset() auront la valeur null.
- Constante insensible à la casse null.
- echo NULL;     => string ("") chaine vide.
- if(NULL) ;     => Bool (false).
- int (NULL+1)   => int (0) + 1 = 1.
- float (NULL+6) => float (0.0) + 6 = 6.
- array          => [] est vide.
```

```
Chaine numériques string  :
-----------------------
* Considérée comme numérique si elle peut interpétér comme int ou float
* Chaînes débutant numériquement. example : "15Youssef" = (int)15.
* SI : String est numérique => int ou float.
* SI : Autorise les chaînes débutant par des caractères numériques => int ou float.
     - Mais un avertissement (Warning) sera émis pour signaler cette conversion implicite.
* SI : string n'est pas numérique, une TypeError est lancée.
```

6. Déclaration Variable :

```
Variables :
-----------
Declaration : $nom_variable = Valeur.

Règles pour les variables PHP :
------------------------------
* une variable commence par le $signe, suivi du nom de la variable [ $nom_variable ]
* Un nom de variable doit commencer par une lettre [a-z] [A-Z] ou Le caractère de soulignement _
* Un nom de variable ne peut pas commencer par un chiffre [0-9]
* Nom ne peut contenir que des caractere alphanimériques et des traits souligenemet (A-a)(0-9)(_).
* Les noms de variables sont sensibles à ma casse ($age  et  $Age sont dex variables différentes).
```

7. Conversion de Type :

```php
# Contrôler l'état d'une variable en PHP

# Déclarée sans attribute valeur.
$a;
echo $a;      # Warning : Undefined variable
var_dump($a); # NULL

echo "\n";

# Pas Déclarée.
echo $b;      # Warning : Undefined Variable
var_dump($b); # NULL

echo "\n";

# Déclarée et assinger valeur.
$c = 17;
echo $c;
var_dump($c);

echo "\n";

# Déclarée et assinger valeur NULL.
$K = NULL;
var_dump($K);

# Unset : Déclarée et assinger valeur.
$d = "Hamza";
var_dump($d);
unset($d);
var_dump($d); // Warning : Undefined Variable => NULL
```

```php
# isset() :  Détermine si une définie et est (différente de null)
# ----------------------------------------------------------------

echo isset($a) ? "True" : "False : Variable sans Valeur\n";          // false
echo isset($b) ? "True" : "False : Variable n'est pas Déclarée\n";   // False
echo isset($c) ? "True : Déclarée avec valeur != NULL \n" : "False"; // True
echo isset($d) ? "True" : "False : Déclarée avec NULL\n";            // False
echo isset($d) ? "True" : "False : Variable a été détruite";         // False

# Variable Définir     : Elle a été déclarée avec une valeur.
# Variable Non Définir : Elle n'est pas déclarée - Déclarée avec sans valeur - Detruire par unset().
# Variable a été détruite avec la fonction unset() : Considérée comme non définie.
# [ Déclarée sans valeur ] - [ Pas déclarée ] - [ unsert() ] = NULL

# 1) isset($var) : True  : Variable Définir avec une valeur differenet NULL.
# 2) isset($var) : False : Variable Non Définir ou Définir Avec une valeur NULL.

# unset() : Pour détruire une variable, la supprimer de la mémoire et la marquer comme non définie.
$test = "Hamza";
unset($test);     // Détruit la variable $nom
var_dump ($test); // Warning : Undefined Variable => NULL

# Var_dump() : afficher des informations détaillées sur une variable, [] type - valeur ]
```

- **Type Juggling et Type Casting :**

  - **Type juggling** : Conversion automatique ou implicite de types de données effectuée par le langage de programmation lui-même. Par le programme.

  - **Type Casting** : (conversion explicite) : convertir de manière explicite un type de données en un autre type spécifié, par développer.

```php
// typecasting : est le processus de modification du type d'une variable d'un type à un autre.
#   Implicite : effectue une conversion de type de manière automatique.
#   Explicite : réalisé de manière explicite par le développeur.

1) Conversion implicite de type avec les opérateurs (+, -, *, /) :

    Des erreurs de type (TypeError) : int/float + [ chaîne non numérique - object - array ]
    Des avertissements (E_WARNING)  : int/float + [ chaîne débutant par un nombre ]
    Pour les Entiers     : octal= int , hexa=int , binaire=int , float=arrondi vers zero, NULL= 0 , false= 0 , true=1.
    Pour les flottants   : NULL= 0 , false= 0 , true=1.
    Pour chaine ou chaine début par numero : convertir en int (Passe PHP_INT_MAX => float) ou float

    NB : Si un des opérandes est un float, les deux opérandes sont interprétés comme des floats, Resulta est Float.

 * Opérateur de concaténation :
    NULL = False = "" ; (int = float) => string ; true = 1
    array() : E_Warning.
    object  : Errore

2) Conversion implicite de type avec des arguments passés à une fonction avec avec un type explicite (int/float) :

    Erreur (TypeError)   : int/float + [ chaîne non numérique - object - array ]
                           [ chaîne non numérique - chaîne débutant par un nombre - NULL ]
    Pour les Entiers     : octal= int , hexa=int , binaire=int , float=arrondi vers zero, false= 0 , true=1.
    Pour les flottants   : false= 0 , true=1.

3) Conversion explicite :

    Casting de type ou fonction(type) : null=0 , false=0 , true=1 , chaine numérique=int/float ,
    Chaine debuatnt numérique = partie numérique (avec E_warning)
    array(vide) =0 , array(non vide) = 1
    Des avertissements (E_WARNING) : Objet sera converti en 1.
```

```
1) Transtypage :
---------------

A) Cast explicite avec l'Opérateur de cast :

* Syntaxe : (type) $variable .
* Type : (int)  (float) (string) (bool) (array) (objet)[stdclass]

NB : Le cast ne modifie pas le type de donnée de la variable d'origine,
     mais crée plutôt une nouvelle valeur convertie.

B) Fonction des cast : à partir n'importe qulle valeur scalare.

* intval()  intval("string",base)  strval() floatval() boolval()

* Nb : intval("0x17") : traiter comme chaine pas comme hexadecimal.

2) Déterminer le type d'une variable en PHP
-------------------------------------------
* gettype($mavar) => retourne string [ type de données de la variable ]
- Type : boolean , Integer , double , array , object , NULL , ressource , string , unknown

Autres fonctions prédéfinies pour connaitre le type d'une variable :
-----------------------------------------------------------
is_numeric ()             : si une variable est un nombre ou une chaîne numérique
is_int () , is_integer()  : si une variable est de type nombre entier
is_double() , is_float()  : si une variable est de type nombre décimal
is_string()               : si une variable est de type chaîne de caractères
is_array()                : si une variable est un tableau
is_object()               : si une variable est de type objet
is_bool()                 : si une variable est un booléen
is_null()                 : si une variable vaut null
is_scalar()               : si une variable est un scalaire [int , float , string , bool]

- NB : declare(strict_types=1); les conversions de type implicites ne sont pas autorisées,
et toutes les conversions doivent être explicites.
```

```
Conversion en Objet :

* Object => Object
* Array => Object(stdClass){ "Key1" => "Val1" , "Key2" => "Val2" }
* Variable (Scalar)  => Object(stdClass) { "Scalar" => Valeur }
* NULL  => Object(StdClass) { }
```

```
Fonctions de Gestion des variables :
------------------------------------

-------------------------------------------------
2) empty() : Détermine si une variable est vide :
-------------------------------------------------

* si elle n'existe pas, ou si sa valeur équivaut :
  [ Null, 0 , false , "" (chaîne vide) , array() (tableau vide) , et 0.0 (flottant) ]

* La fonction ne génère pas d'alerte si la variable n'existe pas.
* empty()    <---->   !isset($var) || $var == false.

5) unset() : pour détruire une variable spécifiée, ce qui libère la mémoire qu'elle occupait.

6) La fonction var_dump() renvoie le type et la valeur du variable :
```

8. Declaration constant :

```
Method define :
--------------
*  Syntaxe      : define(nom, valeur)
*  Régle du Nom : Un nom de constante valide commence par [A-Z][a-z] ou _
*  Contenir des valeurs scalaires (int , float , string , bool) et des tableaux (à partir de PHP 7.0).
*  Les constantes sont automatiquement globales sur l'ensemble du script
*  Les constantes définies à l'intérieur d'une fonction peuvent être utilisées en dehors de la fonction


Mot-clé const :
--------------

* Les constantes doivent être déclarées au niveau supérieur car elles sont définies au moment de la compilation.
* Ils ne peuvent pas être déclarés dans des fonctions, des boucles, des instructions if ou des blocs try/catch.
```

Utilisation dans Classe

* Define : Dynamique est Global, utilisation tous les scripts.
* const : Spécifiques à la classe où elles sont déclarées.
* Constantes magiques : __File__ , __Line__ , __DIR__

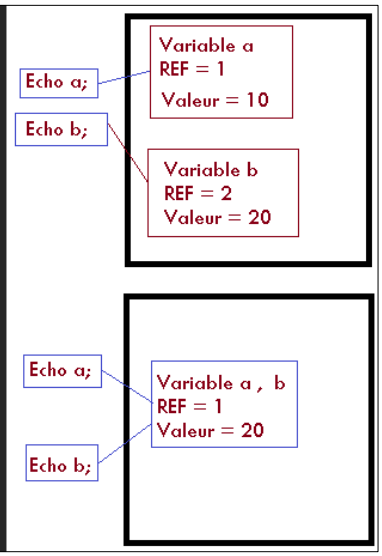
9. Affectation par valeur vs référence :

```
Assignation par valeur  :
-----------------------
$a = 10;
$b = $a;  * Affectation par valeur
$b = 20;  * Modification de $b
echo $a; => 10

* Lorsque vous affectez une variable par valeur, la valeur de $a (10) est
copiée dans $b. Aucun impact sur $a.

Assignation par référence :
-------------------------
$a = 10;
$b = &$a;  * Affectation par valeur
$b = 20;  * Modification de $b
echo $a; => 20

* Lorsque vous affectez une variable par référence, $b est une référence à
$a. Toute modification de $b affecte également la valeur de $a.
```



- Supprimer référence utiliser fonction unset($b).

6) Portée des variables :

- Trois portées de variables : Local, global, static

10. Global :

```
* La portée d'une variable est la partie du script où la variable peut être référencée/utilisée.

* Portée globale :
------------------
   - Une variable déclarée en dehors d'une fonction a une SCOPE GLOBALE et n'est accessible qu'en dehors d'une fonction
   - Une variable qui a une portée globale est accessible globalement.
   - Une variable globale n'est par défaut pas accessible dans un espace local.


1) Accéder à une variable de portée globale depuis un espace local :
---------------------------------------------------------------------
   - Les variables globales sont importées dans le contexte local par référence.
   - Le global mot clé est utilisé pour accéder à une variable globale à partir d'une fonction.
   - PHP stocke toutes les variables globales dans => Variable super globale $GLOBALS $GLOBALS[index]
   - Utiliser la variable super globale $GLOBALS pour accéder localement à de variables de portée globale.
```

11. Local :

```
* Portée locale :
---------------
   - Une variable déclarée dans une fonction a une PORTÉE LOCALE et n'est accessible qu'au sein de cette fonction :
   - Le variable sera par défaut détruite dès la fin de l'exécution de la fonction.


2) Accéder à une variable définie localement depuis l'espace global :
-----------------------------------------------------------------------
   - Il n'y a aucun moyen d'accéder à une variable définie localement depuis l'espace global.
```

12. Static :

```
3) Le mot clef static :
-----------------------
   -  Une variable définie localement va être détruite fin de l'exécution de la fonction.
   -  Une variable locale ne soit PAS supprimée. Conservée pour pouvoir être réutilisée lors d'une prochaine exécution.
```

```php
function tick_handler()
{
    static $i = 0;
    echo "tick_handler() called numero : {$i} \n";
    $i++;
}
```

```
tick_handler() called numero : 0
tick_handler() called numero : 1
tick_handler() called numero : 2
tick_handler() called numero : 3
82tick_handler() called numero : 4
```

7) Commentaire

- Seul ligne : // Commentaire #Commentaire
- Multiple lignes : /* **Lignes**./

8) Les opérateurs :

13. Types des opérateurs :

```
Types les opérateur :
---------------------

1) Unaires :  qui agissent sur une seule opérande. ex:  [ - ! ++ -- ]

2) Binaire : agissent sur deux opérandes. ex :
        Arithmétiques : [ + - . * / %]
        Comparaison   : [ == != < > <= >= <=> ]
        Logiques      : [ && and || or ]
        Affectation   : [ = += -= *= /= |= %= >>= <<= ]
        concaténation : [ . ]

3) ternaire : qui agit sur trois opérandes.
        [ ?: ] [ condition ? valeur_SI_Vrai : Valeur_SI_Faux ]
```

14. Priorité des opérateurs :

```
Priorité des opérateurs :
-------------------------

-------------------------------------------------------------------------
Associative     |     Opérateurs
-------------------------------------------------------------------------
Droite          :  **
N/A             :  + - ++ -- (int) (float) (string) (array) (bool)
N/A             :  !
gauche          :  * / %
gauche          :  + - .
gauche          :  . (string)
N/A             :  < > <= >=
N/A             :  == != === !== <> <=>
gauche          :  &&
gauche          :  ||
droite          :  ?? ( Coalescence NULL )
gauche          :  ?: ( ternaire )
droite          :  = += -= *= **= .= <<= >>= ??=
gauche          :  and
gauche          :  or
-------------------------------------------------------------------------
```

15. Arithmétiques :

```
Opérateurs arithmétiques :
--------------------------

+$a         :   convertir à int ou float
-$a         :   convertir à int ou float
$a + $b     :   Addition
$a - $b     :   Soustraction
$a * $b     :   Multiplication
$a / $b     :   Division
$a % $b     :   Modulus
$a ** $b    :   Exponentionation (Puissance)


// Type juggling :
------------------

// Bool                         :   true(1) , false(0)
// String Numérique             :   "76"(76)
// String Débutant Numérique    :   "45ER"(45)=>(E_Warning)
// Null                         :   0
// Float (Opérateur) Float      :   Le résultat sera toujours (float).
// Array                        :   [Type_Error]
// Object                       :   [Type_Error]
// String Non Numérique         :   [Type_Error]
```

16. Comparaison :

```
* Opérateurs de comparaison *
----------------------------

[ Opérandes (Opérateur) Opérandes ]

* Opérateur [ < > <= >= <=> == === != !== ]

* [ ==  : a   Egal    b ] .  (transtypage). vérifie égalite des valeurs
* [ === : a Identique b ] .  Vérifie égalite valeur et type.


* A <=> B  :  [-1: A<B]  [0: A=B]  [1: A>B]


* Comparaison entre deux chaine de caractére :
-----------------------------------------------

*1) La comparaison est effectuée caractère par caractère,
  en commençant par le premier caractère de chaque chaîne

*2) Les caractères sont comparés en utilisant leur valeur ASCII ou Unicode.

*3) Si les caractères comparés sont identiques, la comparaison passe au caractère suivant

*4) Si les caractères comparés sont différents, la chaîne avec le caractère ayant une valeur ASCII
  ou Unicode inférieure est considérée comme inférieure.

*5) String  +  NULL  : en String null ("")

*6) Fonction strcmp('' , '' ) : [-1 , 0 , 1]
```

```
Résumer :
---------

String  (Opérateur)  String              :  Comparaison Lexicale.
Bool    (Opérateur)  (*)                 :  Comparaison Logique.  True=1 , False=0 ,  '0'  ,  Array(),  NULL ("")
Nombre  (Opérateur)  Nombre              :  Comparaison numérique.
Nombre  (Opérateur)  Chaine Numérique    : Comparaison Numérique; Convertir en Nombre.
Nombre  (Opértaue)   Chaine non Numérique : Comparaison Lexicale. Convertir en String
```

```
-----------------------------------------------------------------------------------------------------------------------
* Opérateurs d'affectation conditionnelle *
-----------------------------------------------------------------------------------------------------------------------

L'opérateur ternaire ?: permet d'effectuer une affectation conditionnelle.
Si l'expression exp est vraie, l'opérande exp_1 est assigné à la variable.
Sinon c'est exp_2 qui est assigné.

Example :
---------
$x = exp ? exp_1 : exp_2;
$x = exp_1. ( SI exp= true  )
$x = exp_2. ( SI exp= false )

-----------------------------------------------------------------------------------------------------------------------

L'opérateur null coalescing ?? permet d'effectuer une affectation conditionnelle basée sur la nullité d'une variable.
Si l'opérande exp1 existe et n'est pas null, il est assigné à la variable.
Sinon c'est exp2 qui est assigné.

Example :
---------
$x = exp1 ?? exp2
$x = exp1 (SI existe , not null)
$x = exp2 (SI exp1 not exist, is null)
```

```
Comparaisons de $x avec des fonctions PHP :
-------------------------------------------

                Empty()         Is_Null()               Isset()
-----------------------------------------------------------------
$x= ""          true            false                   true
$x= Null        true            true                    false
var $x;         true            true                    false
indéfini        true            true                    false
$x = []         true            false                   true
$x = false      true            false                   true
$x = true       false           false                   true
$x = 1          false           false                   true
$x = 42         false           true                    true
$x = 0          true            false                   true
$x = "1"        false           false                   true
$x = "0"        true            false                   true
$x = "PHP"      false           false                   true
$x = "true"     false           false                   true
$x = "false"    false           false                   true


Comparaison large avec ==
-------------------------

$x= ""        : false , null
$x= Null      : false , 0 , [] , "vide"
var $x;       : null , false , 0 , [] , ""
indéfini      : null , false , 0 , [] , ""
$x = []       : false , null
$x = false    : 0 , "0" , null , [] , ""
$x = true     : 1 , -1 , "1" , "-1" , "php"
$x = 1        : true , "1"
$x = 0        : false , 0 , "0" , null
$x = "1"      : true , 1 , "1"
$x = "0"      : false , 0
$x = "PHP"    : true
```

17. Comparaison : String Avec Nombre

```
String  avec Nombre :
---------------------
* Chaine Numérique     : Comparaison numérique. ( String => Nombre (int / float) ).
* Chaine Non Numérique : Comparaison lexicale.  ( Nombre => string  ).
```

18. Opérateurs d'incrémentation et décrémentation

```
Opérateurs d'incrémentation et décrémentation :
-----------------------------------------------
[
   ++$a :       Pre-incrémente  : Incrémente $a de 1, puis retourne $a.
   $a++ :       Post-incrémente : Retourne $a, puis incrémente $a de 1.
   --$a :       Pré-décrémente  : Décrémente $a de 1, puis retourne $a.
   $a-- :       Post-décrémente : Retourne $a, puis décrémente $a de 1.

]

* Chaine non numérique : Incrémentation dérnier caractére de chaine.
* Chaine Alphanémrique : Incrémentation : "A0" => "A1" => .... => "B1"

NB :
    - Les variables de caractères peuvent être incrémentés, mais pas décrémentées
    - Seuls les lettres et chiffres ASCII (a-z, A-Z et 0-9) sont supportés.

* Chaine numérique      : incrémentation/décrémentation nobmre de chaine. => int/float
* nombre                : incrémentation/décrémentation nombre.
* null                  : 1; => int.

* (NB) : Les tableaux, objets, booléen et ressources ne sont pas affectées.

Utilisation Postfix et Prefix :
-------------------------------

* a++;  Use a et change à a+1
* ++a;  Change a+1 et Use a+1
```

```
Utilisation Postfix et Prefix :
-------------------------------

* a++; Variable est d'abord utilisée dans l'expression, puis elle est incrémentée de 1.

* ++a; Variable est incrémentée de 1 d'abord, puis la nouvelle valeur est utilisée dans l'expression.
```

```php
$a = 7;
echo $a++; // 7
echo $a;   // 8

echo --$a; // 7
echo $a;   // 7
```

9) Structures de contrôle :

```php
// Cas Special :
$exp = (10 > 10);

if($exp) echo "true\n";
if(!$exp) echo "False\n";
// Expression  == true  ( Expression )
// Expression ==  false ( !Expression )


// Synatxe :
// ----------
if($exp) echo "True"; # Quand seul instruction par besoin ecrire accolades {}
```

```php
// Synatxe :
// ---------
if($exp) echo "True"; # Quand seul instruction par besoin ecrire accolades {}

// Syntaxe Alternatife :
// ---------------------
# Permet d'intégrer facilement du code PHP dans des fichiers de modèle ou de vue contenant du HTML.
// code PHP contenant des instructions conditionnelles dans le contenu HTML d'une page,

if($exp):
echo "true";
else:
echo "false";
endif;

# for et foreach et while et do-while et switch.
while($exp):
echo "true";
endwhile;

for($j=0;$j<6;$j++):
echo "test";
endfor;
```

```
Switch :
--------
* Instruction
* il ne retourne pas de valeur.
* compare (==)

match :
-------
* Expression
* il retourne une valeur basée sur la correspondance du motif.
* expressions fléchées (=>) pour spécifier les correspondances et les résultats.
* Il ne nécessite pas l'utilisation de l'instruction break car il ne permet qu'une seule correspondance.
* compare avec strict : (===)
```

10) Fonction :

19. Déclaration de Type

Les déclarations de types peuvent être ajoutées aux ***arguments des fonctions***, ***valeurs de retour***, et, à partir de PHP 7.4.0, les ***propriétés de classes***. Elles assurent que les valeurs sont du type spécifié au temps de l'appel, sinon une **TypeError** est **lancée**.

```
Type Nullable :
---------------

* La déclaration des types de paramètre et de valeur de retour peut désormais être marquée en tant que nullable.
* En préfixant le nom du type avec un point d'interrogation.
* le type spécifié aussi bien que null peuvent être passés comme argument, ou retournés en tant que valeur,
respectivement.

Example :
```

```php
//  Paramètres de fonction
2 references
function Test1(?int $Nombre){
    return $Nombre +1;
}
Test1(2); // 3
Test1(NULL); // (int)Null +1  = 0+1
# il peut être Integer ou null.
```

```php
// valeurs de retour
1 reference
function Test2() : ?int {
    return NULL;
}
Test2(); // NULL
# elle peut renvoyer soit un entier, soit null.
```

```
Type mixed :
------------

* Disponible à partir de PHP 8.0.0.
* le type mixed accepte toutes les valeurs.
* [Type union = objet - array  String - float - int - bool - null]
```

```php
// valeurs de retour
1 reference
function Test2($a) : mixed {
    return $a;
}
```

```
Union Type :
------------

* Disponible à partir de PHP 8.0.0
* Type Nullable : String|Null => ?string
* La syntaxe pour déclarer un Union Type utilise le symbole de barre verticale | entre les différents types que
la variable peut prendre.
```

```php
function fonctionUnionType(int|string $parametre) : int|float {
    // Le paramètre $parametre peut être soit un entier, soit une chaîne de caractères
    return 42; // ou 3.14 (un entier ou un nombre à virgule flottante)
}
```

```
PHP Type Hints :
----------------

* PHP est un langage typé dynamiquement.

* Type pour parameters   :
   - Function My_Fonction(type $param1 , type $parm2) { }

* Type pour return value : Function My_Fonction(int $a , int $b) : type { }

* Void type     : Si une fonction ne renvoie pas de valeur, vous utilisez le type void.

* Union type    : Type1 | Type2 | Type3

* Mixed type    : object|resource|array|string|int|float|bool|null

* Nullable type : ?type = Le ?type vous permet de passer valeur en argument ou null.
```

20. Typage Strict

```
Typage Strict  :
---------------

* Par Default, PHP va convertir les valeurs d'un mauvais type vers le type scalaire attendu
tant que possible. Conversion [ Implicite ].

* Active le mode de typage strict fichier par fichier.

* Lorsqu'un fichier PHP contient la déclaration "declare(strict_types=1);" et qu'il est inclus dans un autre
fichier, cette déclaration affecte uniquement le fichier dans lequel elle est déclarée et ne se propage pas aux
fichiers qui l'incluent.

* Debut du Fichier [ declare(strict_type=1)] => Active
```

21. Création Fonction :

```php
echo "\n-----------------------------------------\n";

# Il n'ont pas besoin d'être définies avant d'être utilisées.
Aff1();
$b = 10;
1 reference
function Aff1(){
    global $b;
    echo "b = ". $b;
}

echo "\n-------- Il est définie conditionnellement -----\n";

# Uncaught Error: Call to undefined function Afficher().

$a = "Youssef";
if($a){
    1 reference
    function Afficher(){
        global $a;
        echo "A = ". $a;
    }
}

echo "\n----- Il est définie à l'intérieur d'une autre fonction -----\n";

# Uncaught Error: Call to undefined function Index()
#  Index() n'existe pas tant que  Cherche() n'est pas appelé
Index();
0 references
function Cherche(){
    1 reference
    function Index(){
        echo "Index";
    }
}
```

```
1) Créer une fonction :
   -----------------

Function nomDeLaFonction() {
  // code to be executed;
}

2) Règles pour le nom de la fonction  :
   --------------------------------

- Le nom doit commencer par une lettre ou un souligné (_).
- Les noms ne sont pas sensibles à la casse. [ "maFonction" = "mafonction" = "MaFonction" ].

3) Les fonctions n'ont pas besoin d'être définies avant d'être utilisées, sauf dans deux cas :
   ---------------------------------------------------------------------------------------

* Lorsqu'une fonction est définie conditionnellement, par exemple dans une structure [ "if" ].
* Lorsqu'une fonction est définie à l'intérieur d'une autre fonction
et qu'elle est appelée à partir de la fonction principale.



NB :
----
* Toutes les fonctions et classes en PHP ont une portée globale.
* Elles peuvent être appelées à l'extérieur d'une fonction.
* PHP ne supporte pas la surcharge, la destruction et la redéfinition de fonctions déjà déclarées.
* Il est possible d'appeler des fonctions récursives en PHP.
```

22. Les arguments de fonction :

```
4) Arguments de la fonction PHP :
--------------------------------

Introduction :
--------------

Les informations peuvent être transmises aux fonctions via des arguments.
Un argument est comme une variable.

* Argumenets : Un argument est la valeur réelle passée à une fonction lors de son appel.
* Parametres : Un paramètre est une variable déclarée dans la définition de la fonction qui attend une valeur spécifique lors de son appel.
```

Declaration : **Function** Nom_function(***Paramettres***){ .... }

Appel : Nom_function(***Arguements***).

```
1) Passage d'argumenets par valeur :
------------------------------------

* Par défaut, les arguments sont passés à la fonction par valeur
* un argument est passé à une fonction, une copie de la valeur de cet argument est créée,
  et toute modification effectuée sur cette copie à l'intérieur de la fonction
  n'affecte pas la valeur de l'argument d'origine à l'extérieur de la fonction.
```

```php
$val = 19;
1 reference
function Change($a){
    $a =78;
}
Change($val);    # val = 19
```

2) Passage d'arguments par reference :
---
* vous pouvez passer des arguments par référence en ajoutant le symbole "&" devant l'argument dans la déclaration de la fonction.

```php
$a = 6;
0 references
Function nomDeLaFonction( &$a ) {
  $a = 10;
}
echo $a; // a = 10;
```

NB :
---
* Seules les variables peuvent être passées par référence.
* Les valeurs littérales (constantes) ou les expressions ne peuvent pas être passées par référence.

3) Valeur par défaut des arguments :

Les valeurs par défaut des paramètres peuvent être :

* [des valeurs scalaires : int , bool , string , float , tableaus, null]
* [à partir de PHP 8.1.0, des objets utilisant la syntaxe new ClassName()]
* Les arguments sans valeur par défaut doivent être en premiers.
* les arguments nommés peuvent être utilisées pour passer outre plusieurs paramètres optionnels.

23. Variadic functions

Liste d'argument à nombre variable :

* Le "splat operator" ou "argument unpacking" est représenté par les trois points (...).
* Cette syntaxe permet de déballer (unpack) les éléments d'un tableau et de les utiliser comme des arguments individuels dans une liste d'arguments lors d'un appel de fonction ou de méthode.

```php
#Example 1:

$array = [5,7,9.5];
var_dump(...$array); # int(5) int(7) float(9.5)

# Example 2:
#  Utilisation de ... pour accéder aux arguments variables
1 reference
function Afficher1(...$var){
   $res = 0;
   foreach($var as $val){
       $res += $val;
   }
   return $res;
}
echo Afficher1(3,4,5);

# Utilisation de ... pour fournir des arguments
#Example 3:
1 reference
function Afficher2(...$var){
   $res = 0;
   foreach($var as $val){
       $res += $val;
   }
   return $res;
}
$array = [3,4,5];
echo Afficher2(...$array);

#Example 4:
1 reference
function Afficher3($a,$b,$c){
   return $a+$b+$c;
}
$array = [3,4,5];
echo Afficher3(...$array);
```

```php
# Example 3:
#  Utilisation de ... pour accéder aux arguments variables
1 reference
function Afficher1($a,$b ="z",...$var){
    $res = "";
    foreach($var as $val){
        $res .= $val;
    }
    return $res;
}
# Seul le dernier paramètre peut être variadique [C,D,L,N] .
echo Afficher1("H","B","C","D","L","N");
```

24. Variable Fonction :

```
Function Variable :
-------------------
- Stocker une fonction dans une variable.
- Passer une fonction en tant qu'argument à une autre fonction.
- Assigner une fonction anonyme (fonction sans nom) à une variable.
```

```php
# 1) Stocker une fonction dans une variable.
0 references
function addition(int $a , int $b){
    return $a+$b;
}

$func = "addition";
echo $func(6,7); // 13

# 2) Passer une fonction en tant qu'argument à une autre fonction.
1 reference
function Operation(callable $callback, int $a , int $b){
    return $callback($a,$b);
}

echo Operation("addition",5,4);

# 3) Assigner une fonction anonyme à une variable :
$addition = function ($a, $b) {
    return $a + $b;
};

$result = $addition(5, 3); // Appelle la fonction anonyme pour effectuer l'addition
echo $result; // Output: 8
```

25. Anonymous fonction :

- Return Object de classe **Closure**.

```php
# Anonymous function dans une variable
$Anon_Funct1 = function(){ echo "Hello"; };
echo $Anon_Funct1();


# Anonymous Function avec paramètre dans une variable
$Anon_Funct2 = function($Nom){ echo "Hello {$Nom}"; };
echo $Anon_Funct2("Ahmed");


# Inherit Variable From parent Scope
/* le mot clé "use" est utilisé pour importer des variables
de la portée externe dans une fonction anonyme (fermeture)
dans le contexte de programmation orientée objet */
$msg = "Hello";
$Anon_Funct3 = function($Nom) use ($msg)
{
    echo "$msg $Nom";
};
echo $Anon_Funct3("Ahmed");


# Pass Anonymous Function To Function => array_map
$nums = [10,20,30,40,50];
0 references
function add_five($item){
    return $item + 5;
}


// method 1  :  $nums_after_adding = array_map("add_five",$nums);
// Method 2  :  $nums_after_adding_ten = array_map(function($item){ return $item + 10; },$nums);
echo "<pre>";
print_r($nums_after_adding_ten);
echo "<pre>";
```

26. Arrow Function :

```php
# Arrow Function dans une variable :
# arrow function : fait référence à la flèche (=>)
# arrow function : fn (argument_list) => expr.
# Short syntaxe pour fonction Anonymous function
$var = fn() => "Hello";
echo $var();

# Arrow function avec paramètre dans une variable
$var = fn($Branche) => "Deplom :  $Branche";
echo $var("TDI");

# Arrow  Inherit Variable From parent Scope Is Automaique
$msg = "Diplôme";
$var = fn($Branche) => "$msg $Branche";
echo $var("TDI");

# Pass Arrow Function To Function => array_map
$nums = [10,20,30,40,50];

$num_after_ten = array_map(fn($item) => $item + 10, $nums);
echo "<pre>";
print_r($nums_after_adding_ten);
echo "<pre>";

$var = function(){ echo "Ahmed"; };
var_dump($var);
```

11) ***Déclare :***

```php
// événements de tic seront déclenchés après chaque N=1 instruction exécutée.
declare(ticks=1);

// Cette fonction est appelée à chaque événement de tic.
0 references
function tick_handler()
{
    static $i = 0;
    echo "tick_handler() called numero : {$i} \n";
    $i++;
}

// Un événement qui intervient à chaque tick est spécifié avec la fonction register_tick_function().
register_tick_function('tick_handler');

$a =4;
echo $a;


# Resultat :
#-----------
# tick_handler() called numero : 0
# tick_handler() called numero : 1
#6
#tick_handler() called numero : 2
```

```php
// doit être la toute première déclaration dans le script, avant toute autre instruction.
# pour les valeurs d'arguments et les valeurs de retour
declare(strict_types=1);


1 reference
function add(int $a,int $b): float {
    return ($a+$b);
}

var_dump(add(4,"9")); // Error Strict type
```

12) Include et Require :

- **Insérer** le contenu d'un fichier **PHP** dans un autre fichier **PHP**.
- **Require** : produira une erreur fatale (**E_COMPILE_ERROR**) et **arrêtera** le script
- **Include** : ne produira qu'un avertissement (**E_WARNING**) et le script **continuera**
- **Include_once** et **require_once** : utilisée pour inclure un fichier dans un autre fichier PHP, mais seulement une seule fois.

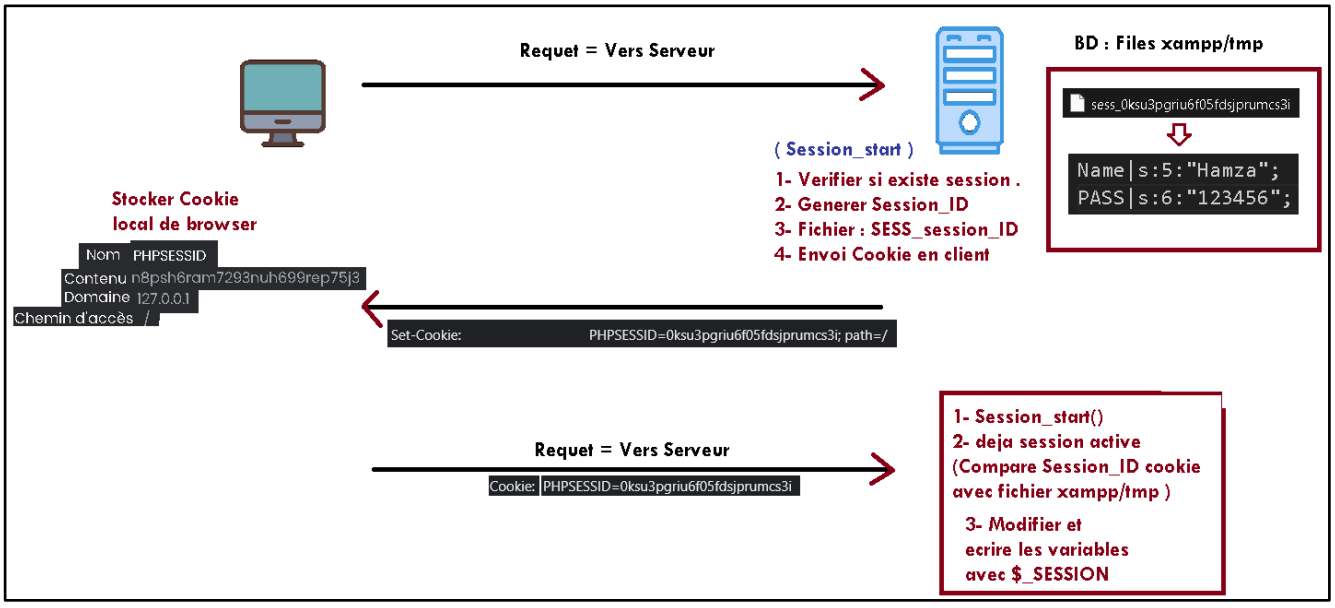
⇒ **Ne** le réinclut **pas**, ce qui peut **éviter** les **problèmes** de **redéfinition** de fonctions ou de variables.

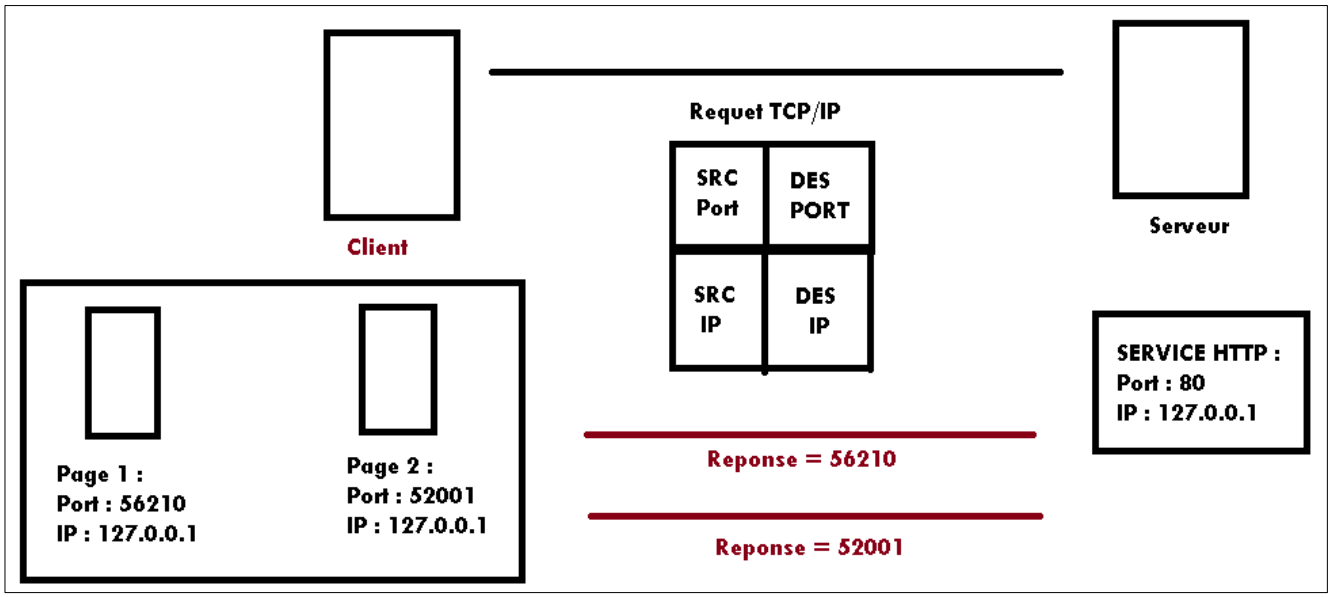
⇒

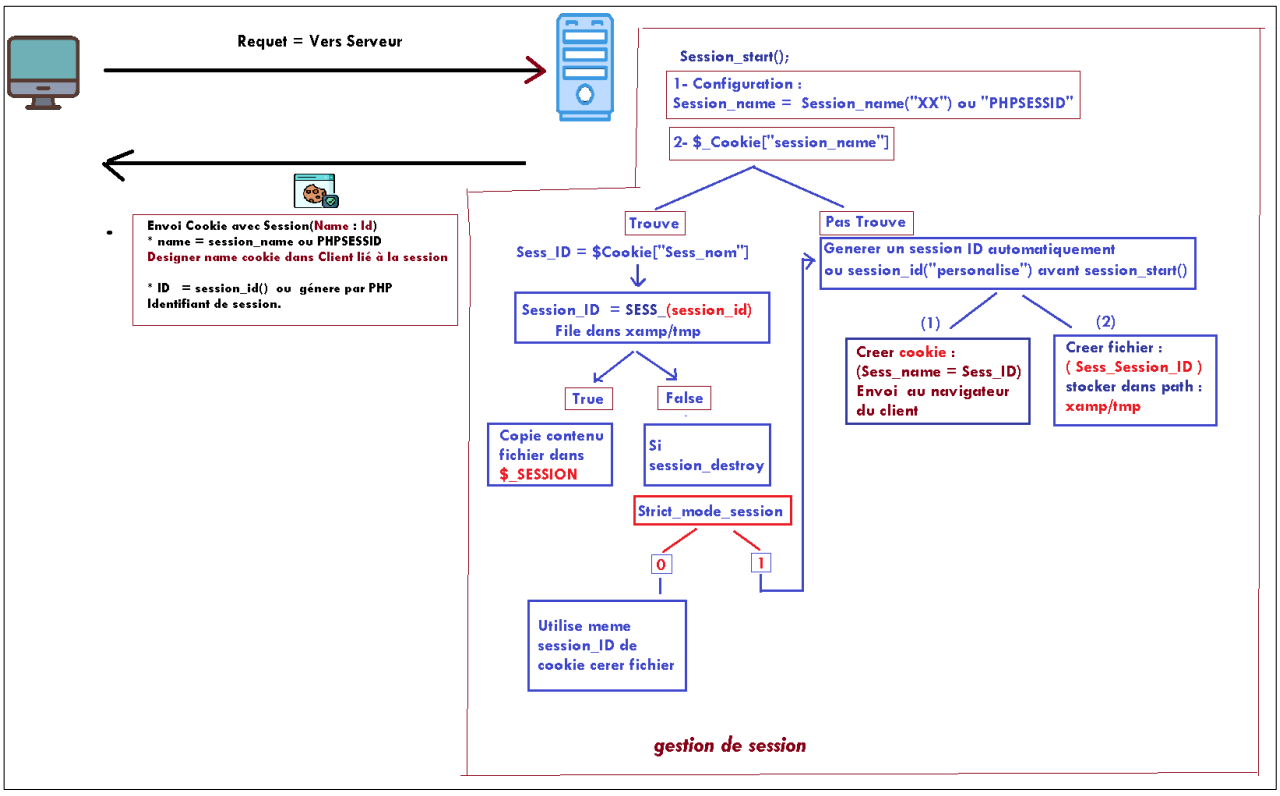
13) Gestion des Sessions :

⇒ Les ports **TCP** et **UDP** sont utilisés pour différencier les applications et les services sur un même hôte (**serveur**) afin de permettre plusieurs connexions simultanées. **Chaque connexion** est **identifiée** par une combinaison d'**adresse IP** (**identifiant de l'hôte**) et de **port** de **transport** (**identifiant de l'application**).

⇒ ***Processus la première foi connecter avec le serveur :***







27. Les fonction Session :

```
session_get_cookie_params :
---------------------------
*  récupérer les paramètres actuels du cookie de session.
*  durée de vie, chemin, domaine, paramètre sécurisé, eparamètre HTTP-only.
*  lifetime - path - domain - secure - httponly
```

```
session_start() : Démarre une nouvelle session ou reprend une session existante
-------------------------------------------------------------------------------

1) Si une session existe déjà pour l'utilisateur
( Cookie : Session_ID == Fichier : SESS_SESSION_ID ) = PHP_SESSION_ACTIVE.

2) Utilise Session_ID via cookie pour récupérer les données de session correspondantes à partir
du fichier de session et les place dans la superglobale $_SESSION.
```

14) Gestion des cookies :

28. Création et Récupération :

```
1) Créatoion d'un cookie et envoi au client :
-----------------------------------------

* setcookie("Name" , "Valeur" , "expires" ,"path")
  - name     : Le nom d'un cookie est soumis aux mêmes règles que les noms des variables.
  - expires : Si aucune valeur n'est passée en argument, le cookie expirera à la fin de la session = (0)
  - path    : Le chemin sur le serveur sur lequel le cookie sera disponible.

Les valeur par default :
-----------------------

* Name   : ("") . Vide.
* Valeur : ("") . Vide.
* expire : (0) . Le cookie expirera à la fin de la session du navigateur.
* Path   : Le chemin relatif à la page actuelle où le cookie est défini.
       localhost/Test_PHP, le chemin par défaut sera /Test_PHP.
* Domaine : Le domaine par défaut est généralement le domaine de l'URL courante.
        http://localhost

NB :  La setcookie()fonction doit apparaître AVANT la balise <html>.

2) Récupérer la valeur d'un cookie :
* Pour récupérer la valeur d'un cookie, nous allons utiliser la variable superglobale $_COOKIE.
```

29. Modification :

```
3) Modifier la valeur d'un cookie :
----------------------------------

appeler à nouveau la fonction setcookie() en lui passant le nom du cookie
dont on souhaite changer la valeur et changer l'argument de type valeur.

NB (Modification) :

* Valeur , Expire :
  - vous n'avez pas besoin de créer un nouveau cookie distinct.
  - Le navigateur comprendra qu'il s'agit d'une mise à jour du cookie existant avec des valeurs modifiées.


* Path , Domaine :
  - le navigateur considérera cela comme la création d'un nouveau cookie distinct.
  - Le cookie précédent avec l'ancien chemin ne sera pas écrasé et restera inchangé.

* Secure ou HttpOnly :
- Pour modifier la propriété "Secure" ou "HttpOnly" d'un cookie existant,
   - vous devez créer un nouveau cookie avec les nouvelles propriétés.
```

30. Supprime :

```
4) supprimer un cookie :
------------------------
* Si la nouvelle date est dans le passé, le navigateur supprimera le cookie, le considérant comme expiré.

a) premier method :
-------------------
* Appeler setcookie() en lui passant le nom du cookie qu'on souhaite supprimer.
* Allons cette fois-ci définir une date d'expiration se situant dans le passé pour le cookie en question.
time() - 3600;

b) Deuxieme method :
--------------------
* setcookie("nom_cookie")
* utilisez la fonction setcookie() avec une seule argument, comme dans votre exemple setcookie("Test");,
  le cookie spécifié est supprimé du navigateur.
* valeur = deleted   et   expire = 0

// Suppression de la valeur du tableau $_COOKIE
  unset($_COOKIE['designPrefere']);
```

15) MySQLI Database :

1. MYSQLI se connecte à MySQL :

```php
// 1- Se connecter à MySQL.
$connexion = new mysqli("localhost", "root", "Aa@123456");
if(!$connexion->connect_errno){
    echo "Connexion Avec Success!!<br><br>";
}
else {
    die ("Erreur : " . $connexion->connect_error);
}
```

```
Extension MySQLi :
------------------

L'extension MySQLi (améliorée) est une manière puissante et flexible d'interagir avec les bases de données MySQL en PHP.
Elle offre deux styles d'API, orienté objet et procédural, pour effectuer diverses opérations de base de données. Dans ce cours,
nous explorerons en détail comment utiliser MySQLi pour interagir avec MySQL.


1) MYSQLI :
-----------
* MySQLi est une extension de PHP spécifiquement conçue pour interagir avec des bases de données MySQL.
* Elle offre une interface (API) pour effectuer des opérations de base de données.
- Orientée objet.
- Procédurale.

* La classe mysqli en PHP représente une connexion entre PHP et une base de données MySQL.

1) Ouvre une connexion à un serveur MySQL
-----------------------------------------
* Mysqli::__construct   : mysqli_connect : Ouvre une connexion à un serveur MySQL
* Mysqli::__construct() : Retourne toujours un objet qui représente la connexion au serveur MySQL.
* Mysqli_connect        : Retourne un objet qui représente la connexion au serveur MYSQL, false si une erreur survient.
* Example :   $con = new mysqli(Hostname, username, password, databse = "") : mysqli | false


2) Sélectionner une Base de Données Spécifique :
------------------------------------------------
*  Mysqli::select_db - Mysqli_select_db : permet de sélectionner une base de données par défaut pour les requêtes.
* 'DB' est le nom de la base de données que vous souhaitez sélectionner.
*  Elle retourne true en cas de succès ou false en cas d'échec.


3) Fermeture De Connexion Dans MySQLi  :
---------------------------------------
* Utiliser la method du class MYSQLI :   Mysqli::close().
```

2. Create Database :

```php
// 2- Création d'une base de données.
$connexion->query("CREATE DATABASE IF NOT EXISTS test_1");
if ($connexion->errno) { die("Erreur : " . $connexion->error); }
else { echo "Création de la base de données réussie!!!!<br><br>"; }

// 2- Sélectionne une base de données par défaut pour les requêtes
$connexion->select_db("test_1");
```

3. Create Tabale :

```php
// 3- Création d'une table.
$res = $connexion->query("CREATE TABLE IF NOT EXISTS  post(
    Id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    Nom VARCHAR(30),
    Prenom VARCHAR(30))");
if ($res) {  echo "Table créée avec succès!!!<br><br>";  }
```

4. Gestion des erreurs:

```
Gestion des erreurs
-------------------
* mysqli::$connect_errno : Retourne le code d'erreur du dernier appel de connexion, 0 aucune erreur n'est survenue.
* mysqli::$connect_error : Retourne le message d'erreur  de la dernière erreur de connexion. null si aucune erreur ne survient.
* mysqli::$error         : Retourne le message d'erreur de la dernier appel à une fonction MySQLi. Une chaîne de caractères vide si aucune erreur.
* mysqli_errno           : Retourne le code erreur pour le dernier appel à une fonction MySQLi. 0 signifie qu'aucune erreur n'est survenue.
```

5. Exécuter des requêtes SQL

```
-----------------
 mysqli::query
-----------------
 *  Exécute une requête sur la base de données.
 *  mysqli::query(string $query) - mysqli_query(mysqli $mysql, string $query) :  ($result_mode = MYSQLI_STORE_RESULT).
 *  Elle retourne false en cas d'échec de la requête.
 *  Pour des requêtes réussies [ SELECT ], mysqli_query() retournera un objet mysqli_result.
 *  Pour les autres types de requêtes réussies (comme INSERT, UPDATE, DELETE), mysqli_query() retournera true.

-------------------------
 Mysqli::execute_query
-------------------------
 * La méthode mysqli::execute_query() est un raccourci pour ( prepare(), bind_param(), execute() et get_result() )
 * Prépare la requête SQL, lie les paramètres et l'exécute.
 * mysqli::execute_query(string $query, ?array $params = null): mysqli_result|bool
 * Le modèle d'instruction peut contenir zéro ou plusieurs marqueurs de paramètres de point d'interrogation (?).
 * Les valeurs des paramètres doivent être fournies sous forme de tableau à l'aide du paramètre params.
 * Renvoie false en cas d'échec. Pour les requêtes ( SELECT ) renvoie un objet mysqli_result. les autres renvoie vrai.
 * Nombre de lignes affectées : $affected_rows. -- Nombre de colonnes : $field_count
```

6. Insertion de données dans des tables :

a. Query et execute query :

```
? Mysqli::insert_id
--------------------
* Récupérer la valeur générée pour une colonne AUTO_INCREMENT de la dernière requête INSERT ou UPDATE.
* la fonction retournera la première valeur AUTO_INCREMENT générée.
* La fonction retourne 0 si la requête précédente n'a pas modifié la valeur AUTO_INCREMENT.
* Appeler immédiatement après l'exécution de la requête qui génère la valeur, sinon la valeur peut être perdue.
* Seul les requêtes émises par la connexion courante affecte la valeur de retour.
```

```php
# 1 ) Avec Method Query.

$sql = "INSERT INTO post(Nom,Prenom) values('Hamza','El-Khanchoufi')";
if($connexion->query($sql)) {  echo "Ajouter Bien Sucess !!!<br><br>";  }

# 2) Avec Method execute_query.

$sql = "INSERT INTO post(Nom,Prenom) values('Youssef','El-Khanchoufi')";
if($connexion->execute_query($sql)) {  echo "Ajouter Bien sucess !!!<br><br>";  }

# 3) Avec Method execute_query avec Array.

$sql = "INSERT INTO post(Nom,Prenom) values(?,?)";
if($connexion->execute_query($sql,['Hamza','El-khanchoufi'])) {  echo "Ajouter Bien sucess !!!<br><br>";  }

# Mysqli::insert_id : Récupérer la valeur générée pour une colonne AUTO_INCREMENT.
echo "\nID de la dérnier requet excuter est : " . $connexion->insert_id ."\n";
```

7. Les Requetés préparées :

```
Utilisation de mysqli_stmt (Requêtes préparées) :
-------------------------------------------------
* La classe mysqli_stmt en PHP est utilisée pour représenter une requête préparée dans MySQL.

Prepare les Requets SQL:
------------------------
* Mysqli::prepare(query) ou mysqli_prepare(mysqli , query) :  Prépare une requête SQL pour l'exécution.  mysqli_stmt|false.
* La requête doit être composée d'une seule requête SQL.
* Elle  peut contenir des paramètres de marques (signe '?') à remplacer par des valeurs lors de l'exécution.

Exécuter la requete Préparer SQL :
----------------------------------
* mysqli_stmt::execute(Array) ou mysqli_stmt_execute(mysqli_sqtm , Array) :  exécuter une requête préparée avec MySQLi en PHP.
* Cette fonction retourne true en cas de succès ou false si une erreur survient.

Get_result :
------------
* Récupère un jeu de résultats d'une déclaration préparée sous la forme d'un objet mysqli_result.
* Mysqli_stmt::get_result(): mysqli_result | false
* Cette méthode ne doit être appelée que pour les requêtes qui produisent un ensemble de résultats. (Select)
* Les données seront récupérées depuis le serveur MySQL vers PHP.
* Disponible uniquement avec mysqlnd.
* False pour d'autres requêtes DML ou en cas d'échec.

Fetch :
------
* Retourne le résultat d'une requête préparée dans une variable, liée par mysqli_stmt_bind_result().
* Notez que toutes les colonnes doivent être liées par l'application avant d'appeler mysqli_stmt_fetch().
* True : Réussite. Les données ont été lues. | False : Une erreur est survenue. | Null : Il n'y a plus de ligne à lire ou les données ont été tronquées.

bind_result :
-------------
*  mysqli_stmt::bind_result(mixed &$var, mixed &...$vars): bool
* Lorsque mysqli_stmt_fetch() est appelée pour lire des valeurs, place les données dans les variables spécifiées var/vars.
* Cette fonction retourne true en cas de succès ou false si une erreur survient.
```

```
Insert_id :
----------------
* Récupère l'ID généré par la dernière requête INSERT

Affected_rows :
---------------
* Retourne le nombre total de lignes (Update, Insert, Delete ) par la dernière requête.
* -1 : la requête a retourné une erreur ou que, pour une requête SELECT.

Num_rows :
----------
* Retourne le nombre de lignes extraites du serveur (Select).
* Il ne fonctionnera qu'après l'appel de mysqli_stmt_store_result().
* Il retourne 0 à moins que toutes les lignes aient été récupérées du serveur.

Mysqli_stmt::$param_count :
---------------------------
* mysqli_stmt_param_count – Retourne le nombre de paramètres d'une commande SQL
```

b. Avec execute (array) avec un tableau de valeurs :

```php
# 3) Avec method mysqli_stm::executer(array) avec un tableau de valeurs.

$sql = "INSERT INTO post(NOM,Prenom) values(?,?),(?,?)";
$stm = $connexion->prepare($sql);
if($stm->execute(["Anwar","jadour","Ilyas","Real"])) {
    echo "<br>Post N° :". $stm->insert_id ." Ajoute Bien Sucess!!!!<br>";
    echo "Nombre de lignes Ajouter :".  $stm->affected_rows ."<br>";
}
```

c. Avec execute (array) en utilisant BindParam() :

```php
# 4) Avec Method mysqli::stm_execute en utilisant bind_param().

$sql = "INSERT INTO post(NOM,Prenom) values(?,?)";
$stm = $connexion->prepare($sql);
$Nom = "A/C Sabir"; $prenom = "El-Khanchoufi";
$stm->bind_param("ss",$Nom,$prenom);
if($stm->execute()) { echo "Ajouter Bien Sucess!!!!<br><br>"; }
```

8. Récupérer des données de la base de données :

d. Sélection Avec query et execute_query :

```
Query :
-------
* Mysqli::query( Query , Result_Mode = Mysqli_Store_Result ) : mysqli_result|Bool
* query (select) = Mysqli_real_query + mysqli_store_result
* Renvoi result mis en mémoire tampon

Executer_Query :
---------------
* execute_query( query , array(param) ) mysqli_result | bool
*  prepare() + mysqli_stmt::bind_param() + mysqli_stmt::execute() + mysqli_stmt::get_result()
```

```php
#1- Récuperer les données avec la méthode query()
$sql = "select * from post";
$resul = $connexion->query($sql);
print_r( $resul->fetch_all(MYSQLI_ASSOC));


#2- Récuperer les données avec la méthode executer_query( Mysqli , Array)
$sql = "select * from post where NOM=?";
$result = $connexion->execute_query($sql,["Hamza"]);
print_r( $result->fetch_all(MYSQLI_ASSOC));
```

e. Sélection Avec (requête prépare) = prépare + exécuter :

⇒ **Récupération des résultats en utilisant l'interface mysqli_result avec (get_result)**

```php
$stm = $connexion->prepare("Select * from post");
$stm->execute();
$result = $stm->get_result();
echo " Nombre de champs  dans jeu de resultat est : ". $stm->field_count."\n";
echo " Nombre de Lignes dans jeu de resultat est : ". $stm->num_rows."\n";
print_r($result->fetch_all());
```

⇒ ***Récupération des résultats en utilisant des variables liées (bind_result) et (fetch)***

```php
$stm->execute();
$stm->bind_result($Id,$Nom,$Prenom);
while($row = $stm->fetch()){
    echo "ID : ". $Id ." Nom :".$Nom . " prenom : ". $prenom."\n\n";
}
```

⇒ ***Filed count et num_rows avec Requete prepare utilise store_result()***

```php
# filed_count et num_rows avec Requete prepare utilise store_result()
$stm->execute();
$stm->store_result();
echo " Nombre de champs  dans jeu de resultat est : ". $stm->field_count."\n";
echo " Nombre de Lignes dans jeu de resultat est : ". $stm->num_rows."\n";
```

f. Method des récupérations des données :

```php
# Fetch_row : Index (Num)
$result = $connexion->query("select * from post");
echo "\n----------- Fetch_row ---------\n";
while($row = $result->fetch_row()){
    echo "ID : {$row[0]} - Nom : {$row[1]} - Prenom : {$row[2]} \n";
}

# Fetch_column
$result = $connexion->query("select * from post");
echo "\n-----------Fetch_column ---------\n";
while($row = $result->fetch_column(1)){
    echo "Nom : $row \n";
}

# Fetch_object
0 references | 0 implementations
class post{ }
$result = $connexion->query("select * from post");
echo "\n-----------Fetch Object---------\n";
print_r($result->fetch_object("post"));
while($row = $result->fetch_object("post") ){
    echo "ID : ". $row->Id ." - Nom : ".$row->Nom. " - Prenom : ".$row->Prenom. "\n";
}
```

```php
#Fetch_ALL
$result = $connexion->execute_query("select * from post where NOM=?",["Hamza"]);
print_r( $result->fetch_all(MYSQLI_ASSOC));

# Fetch_assoc (associatif)
$result = $connexion->query("select * from post");
echo "\n----------- Fetch_assoc ---------\n";
while($row = $result->fetch_assoc()){
    echo "Id : {$row['Id']} - Nom : {$row['Nom']} - Prenom :  {$row['Prenom']} \n ";
}

# Fetch_array (BOTH = Associative + numérique)
$result =$connexion->query("select * from post");
echo "\n----------- Fetch_array ---------\n";
print_r($result->fetch_array());
while($row = $result->fetch_array(MYSQLI_ASSOC) ){
    print_r($row);
    echo "ID : {$row['Id']} - Nom : {$row['Nom']} - Prenom : {$row['Prenom']} \n";
}
```

```
--------------------------------------------------------------------------
 ! mysqli_result : Représente le jeu de résultats obtenu depuis une requête
--------------------------------------------------------------------------

---------------------------------------------------------------
 ! Les methods de Récuperer des données d'un jeu de résultats
---------------------------------------------------------------

Fetch_All(Mode) : Récupère un tableau deux dimensionnel de toutes les lignes de résultats
dans un tableau associatif, numérique, ou les deux

--------------------
 ? Fetch_array(Mode)
--------------------
* Récupère la ligne suivante d'un ensemble de résultats sous forme de tableau associatif, numérique ou les deux
* Chaque appel ultérieur à cette fonction renverra la ligne suivante, ou null s'il n'y a plus de lignes.
* Si plusieurs colonnes ont le même nom, la dernière colonne écrasera les données précédentes.
* Retourne null s'il n'y a plus de lignes - Retourne false en cas d'erreur.
* Mode : ( MYSQLI_ASSOC :  associatif )  ( MYSQLI_NUM :  numérique ) ( MYSQLI_BOTH : à la fois associatif et numérique ).

--------------
 ? Fetch_assoc
--------------
* Récupère la ligne suivante d'un ensemble de résultats sous forme de tableau associatif
* Chaque appel ultérieur à cette fonction renverra la ligne suivante, ou null s'il n'y a plus de lignes.
* Chaque clé du tableau représente le nom d'une des colonnes du jeu de résultats.
* Renvoie null s'il n'y a plus de lignes dans le jeu de résultats.
* Renvoie false en cas d'erreur.

---------------
 ? Fetch_column
---------------
*  récupérer une colonne unique de la prochaine ligne d'un jeu de résultats (result set) provenant d'une requête MySQL.
*  mysqli_result::fetch_column(int $column = 0) : null|int|float|string|false
*  Les appels ultérieurs renvoient les valeurs de la colonne suivante dans le jeu de résultats, ou false s'il n'y a plus de lignes.
*  La fonction définit les champs NULL à la valeur PHP null lors de la récupération de données.
```

```
-------------
 ? Fetch_row
-------------
* Récupère une ligne de résultat sous forme de tableau indexé
* Chaque nouvel appel retournera la prochaine ligne dans le jeu de résultats, ou null s'il n'y a plus de lignes.
* Cette fonction définit les champs NULL à la valeur PHP null.
* false si une erreur survient.

----------------
 ? Fetch_Object
----------------
* Retourne la ligne suivante d'un ensemble de résultats sous forme d'objet
* mysqli_result::fetch_object(string $class = "stdClass")
* chaque propriété représente le nom de la colonne du jeu de résultats.
* Class : Le nom de la classe à instancier. Si non fourni, un objet stdClass sera retourné.
* Retourn : null s'il n'y a plus de lignes dans le jeu de résultats, ou false si une erreur survient.
```

9. Récuperer les informations sur les conoles d'un jeu de résultats

```php
# ------------------------------------------------------------  #
# Récuperer les informations sur les conoles d'un jeu de résultats #
# ------------------------------------------------------------ #

# Fetch_fields :
$res = $connexion->query("select * from post");
$array = $res->fetch_fields();
foreach($array as $field ){
    echo "Name : ". $field->name . " - type :  " . $field->type . " - Length : " . $field->length . "\n";
}

# Fetch_field :
echo "\n\n--- Fetch field ---\n\n";
while($colone = $res->fetch_field()){
    echo "Colone N° : ". $res->current_field."\n";
    print_r($colone);
}

# Fetch_field_direct :
echo "\n\n ------- Fetch field direct ------- \n\n";
print_r($res->fetch_field_direct(1)); echo "\n";
echo "Position current : ". $res->current_field . "\n";


# Deplacer poiture vers colone 2.
echo "\n\n---------- Fetch seek -----------\n\n";
echo "Colone N° : ". $res->current_field."\n";
$res->field_seek(0);
echo "Colone N° : ". $res->current_field."\n";
print_r( $res->fetch_field() );
echo "Colone N° : ". $res->current_field."\n";
print_r( $res->fetch_field() );
```

```
-------------------------------------------------------------------------------
# Les methods de Récuperer les informations sur les conoles d'un jeu de résultats #
-------------------------------------------------------------------------------

? mysqli_result::$field_count - mysqli_num_fields : Récupère le nombre de champs dans l'ensemble de résultats.

---------------------------
  ? mysqli_result::$lengths
---------------------------
* Retourne la longueur des colonnes de la ligne courante du jeu de résultats.
* Un tableau d'entiers représentant la longueur de chaque colonne (sans inclure les caractères null de fin).
* Retourne False en cas d'erreur.
* Retourne False : Appelée avant les fonctions mysqli_fetch_row(), mysqli_fetch_array(), mysqli_fetch_object().
* Retourne False : Appellée après avoir récupéré toutes les lignes du résultat.

------------------------------
 ? mysqli_result::fetch_fields
------------------------------
* Récupère les informations sur toutes les colonnes du jeu de résultats.
* Retourne un tableau d'objets mysqli_field représentant chaque colonne.
* Propriétés de l'objet : name - table - lenght - type

------------------------------
 ? mysqli_result::fetch_field
------------------------------
* Récupère les informations sur la colonne suivante du jeu de résultats.
* Retourne un objet mysqli_field représentant la colonne.
* obtenir des informations sur chaque colonne du jeu de résultats dans une boucle.
* false si aucune information n'est disponible pour ce champs.

---------------------
 ? fetch_field_direct
---------------------
* mysqli_result::fetch_field_direct(int $index) : object | false
* Retourne un objet qui contient les métadonnées d'un champ dans le jeu de résultats spécifié.
* Le numéro du champ. Cette valeur doit être dans l'intervalle 0 à nombre de champs - 1.
```

```
-------------
 ? filed_seek
-------------
* public mysqli_result::field_seek(int $index): bool
* déplace le pointeur de résultat (curseur) sur le champ spécifié dans un jeu de résultats MySQLi.
* Le prochain appel à la fonction mysqli_fetch_field() retournera la définition du champ de la colonne associée à cette position.
* Index : Le numéro du champ. Cette valeur doit être dans l'intervalle 0 à nombre de champs - 1.
* Retourne toujours true.

-----------------
 ? current_field
-----------------
* Récupère la position courante d'un champ dans un pointeur de résultat
* int $mysqli_result->current_field;
* Retourne la position du curseur de champ utilisé par le dernier appel à la fonction mysqli_fetch_field().
* Cette valeur peut être utilisée comme argument de la fonction mysqli_field_seek().
```

```php
# Les Autres Fonctions :
$res = $connexion->query("select * from post");
echo " Nombre de champs  dans jeu de resultat est : ". $res->field_count."\n";
echo " Nombre de Lignes dans jeu de resultat est : ". $res->num_rows."\n";
$row = $res->fetch_assoc();
foreach( $res->lengths as $i => $val ){
    printf("Le champ n°%2d a une longueur de %2d\n", $i+1, $val);
}
```

```
* mysqli_result::$num_rows -- mysqli_num_rows — Retourne le nombre de lignes dans le jeu de résultats
* mysqli_result::$field_count -- mysqli_num_fields — Récupère le nombre de champs dans l'ensemble de résultats : int
* Un objet mysqli_result retourné par mysqli_query(), mysqli_store_result(), mysqli_use_result(), ou mysqli_stmt_get_result().
```

10. Exécute une ou plusieurs requêtes sur la base de donnée

```php
# Exécution de requêtes multiples (select + Insert) avec multi_query
echo "\n\n------------ Multi Query -----------------<pre>\n\n";

$sql = "SELECT * FROM post; INSERT INTO post(Nom,Prenom) VALUES('Hamza','El-Khanchoufi')";

if ($connexion->multi_query($sql)) {
    do {
        if ($connexion->field_count) {
            // Il y a des résultats à récupérer (requête SELECT)
            $result = $connexion->store_result();
            while ($row = $result->fetch_assoc()) {
                // Traitez les résultats de la requête SELECT ici
                echo "ID : " . $row["Id"] . " Nom : " . $row["Nom"] . " Prénom : " . $row["Prenom"] . "<br>";
            }
            $result->free();
        } else {
            // Pas de résultats à récupérer (requête INSERT)
            $affected_rows = $connexion->affected_rows;
            if ($affected_rows > 0) {
                echo "Nombre de lignes insérées : " . $affected_rows . "<br>";
            }
        }
        echo "More result : ". $connexion->more_results() ."\n";
    } while ($connexion->next_result());
}
```

```
Mysqli::multi_query - Mysqli_multi_query
----------------------------------------
* Exécute une ou plusieurs requêtes, rassemblées dans le paramètre query par des points-virgules.
* Mysqli::multi_query(string $query) : bool
* Mysqli_multi_query() attend pour la première requête de compléter avant de retourner le contrôle à PHP.
* utiliser une do-while pour traiter plusieurs requêtes.
* Pour traiter la prochaine requête dans la suite, utiliser mysqli_next_result().
* Pour vérifier s'il y a plus de résultats, utiliser mysqli_more_results().
* Pour les requêtes SELECT , mysqli_use_result() ou mysqli_store_result() peut être utilisé pour récupérer le jeu de résultat.
* Pour les autres les requêtes, les mêmes fonctions utilisé pour récupérer les informations tel que le nombre de ligne affectés.
* La connexion sera occupé jusqu'à ce que toutes les requêtes soit complété et que leur résultat soit récupéré par PHP.

Mysqli_more_results()
---------------------
* Mysqli::more_results - mysqli_more_results : Vérifie s'il y a d'autres jeux de résultats MySQL disponibles
* Public mysqli::More_results() : bool
* Indique si un ou plusieurs jeux de résultats sont disponibles, générés par un appel antérieur à mysqli_multi_query().
* True si un ou plusieurs jeux de résultats (incluant les erreurs) sont disponibles à partir d'un précédent appel à la fonction mysqli_multi_query().
* False sinon.


Mysqli_next_result()
--------------------
* Mysqli::next_result : Prépare le prochain résultat d'une requête multiple, initialisé par un appel antérieur à mysqli_multi_query(),
* True en cas de succès.
* False si une erreur survient. false si la prochaine déclaration résulte en une erreur, pas comme mysqli_more_results().


Mysqli::Mysqli_store_result()
-----------------------------
* Mysqli::store_result -- mysqli_store_result : Transfère un jeu de résultats à partir de la dernière requête.
* Mysqli::store_result(int $mode = 0) : mysqli_result|false
* Retourne un résultat stocké sous la forme d'un objet ou false si une erreur survient.
* utilisée pour stocker le résultat de la requête actuelle après l'exécution de multi_query.
* Elle renvoie un objet mysqli_result contenant les données de la requête.
* Dans la boucle while, les données sont extraites et affichées pour chaque ligne de résultat.
```

11. Gestioner des errores et Exceptions.

```
Erreurs / Exceptions :
----------------------
* Si le rapport d'erreurs mysqli est activé (MYSQLI_REPORT_ERROR) et que l'opération demandée échoue, (Warning) est généré.
* Si, en plus, le mode est défini sur MYSQLI_REPORT_STRICT, une mysqli_sql_exception est lancée à la place.


Mode de Rapport d'Erreur :
--------------------------

Définition : Le mode de rapport d'erreur dans MySQLi détermine comment les erreurs sont gérées et rapportées
lors de l'exécution de requêtes et d'autres opérations de base de données.

- Mode error : Exception ou Avertissemenet.
- Style procédural : mysqli_report() : Définit le mode du rapport d'erreur mysqli.
- MYSQLI_REPORT_ERROR   : Rapporte les erreurs depuis les appels aux fonctions mysqli.
- MYSQLI_REPORT_STRICT  : Lance une exception mysqli_sql_exception pour les erreurs.
- MYSQLI_REPORT_OFF avec error_reporting(0);    : Désactive les rapports.
- Par default           : MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT

- mysqli_sql_exception :
-------------------------
- Propriétés : ( Message | Code | File | Line | trace ).
- Méthodes   : ( getSqlState() ).
```

12. Insérer Multiple Lignes :

```php
$sql = "INSERT INTO post(ID,Nom,Prenom) Values(?,?,?),(?,?,?)";
$stm = $connexion->prepare($sql);
$stm->bind_param("ississ",$ID,$NOM,$PRENOM,$ID1,$NOM1,$PRENOM1);

$ID=210;
$NOM="Hamza";
$Prenom = "El-Khanchoufi";

$ID1=211;
$NOM1="Anwar";
$PRENOM1 = "El-Kalloubi";
if($stm->execute()) {echo "Ajouter Bien Succéss<br>";};

$stm->close();
```

13. Update Lignes :

```php
// MYSQLI Orientée objet :
$connexion = new mysqli();
$connexion->connect("localhost","root","Aa@123456","test_1");

 $sql = "UPDATE post SET  Nom = ?, Prenom = ? WHERE  ID = ?";
 $stm = $connexion->prepare($sql);
 if($stm){
    $stm->bind_param("ssi",$NOM,$PRENOM,$ID);
    $ID=3;
    $NOM="Hamza";
    $PRENOM = "El-Khanchoufi";
    if($stm->execute()) {echo "Modifier Bien Succéss<br>";};
 }
 $stm->close();
```

```php
// MySQLi Procedural :
$connexion = mysqli_connect("localhost","root","Aa@123456","test_1");
if(mysqli_connect_errno()){ die("Errore : ". mysqli_connect_error()) ; }

$stm = mysqli_prepare($connexion,"UPDATE post SET Nom=? , Prenom=? WHERE ID=?");
if($stm){
 mysqli_stmt_bind_param($stm,"ssi",$Nom,$PRENOM,$ID);
 $ID="5";
 $Nom="Youssef";
 $PRENOM="El-Khanchoufi";
 if(mysqli_stmt_execute($stm)){ echo "Modifier Bien Succéss<br>"; };
}
mysqli_stmt_close($stm);
```

14. Delete Ligne :

```php
// MYSQLI Orientée objet :
 $sql = "DELETE FROM post WHERE  ID = ?";
 $stm = $connexion->prepare($sql);
 if($stm){
    $stm->bind_param("i",$ID);
    $ID=3;
    if($stm->execute()) {echo "Suprimer Bien Succéss<br>";};
 }
 $stm->close();
```

```php
// MySQLi Procedural :
$stm = mysqli_prepare($connexion,"DELETE FROM post WHERE  ID = ?");
if($stm){
 mysqli_stmt_bind_param($stm,"ssi",$Nom,$PRENOM,$ID);
 $ID="5";
 $Nom="Youssef";
 $PRENOM="El-Khanchoufi";
 if(mysqli_stmt_execute($stm)){ echo "Modifier Bien Succéss<br>"; };
}
mysqli_stmt_close($stm);
```

16) PDO :

15. PHP se connecte à MySQL :

```
PDO : extension PHP ("PHP Data Objects").

1) PDO::__construct :
---------------------
* Cette méthode permet de créer une instance PDO qui représente une connexion à une base de données.
* Example           : $conx = new PDO($dsn , $username , $password).

DSN :
-----
* Chaîne de connexion spécifiant le type de base de données, l'hôte, le nom de la base de données et d'autres paramètres.
* Syntaxe : [ $dsn =  "driver:host=hostname;dbname=database" ]

* driver            : Le nom du pilote de BD => (ex: "mysql" = MySQL, "pgsql" = PostgreSQL, "sqlite" = SQLite).
* host = hostname   : L'adresse de l'hôte où se trouve la base de données.
* dbname = database : Le nom de la base de données à laquelle vous voulez vous connecter.


$connexion = new pdo($dsn,$user,$pass);    =>     return object (PDO)

Note : La création de l'instance PDO lèvera toujours une exception de type PDOException si la connexion échoue.
```

```
Fermer la connexion PDO :
-------------------------
* Utiliser la méthode null pour libérer la référence à l'objet PDO.
* Example :   $connexion = null..
```

```php
$host = "localhost";
$db   = "test_1";
$user = "root";
$pass = "Aa@123456";
// DSN :
$dsn = "mysql:host=".$host.";dbname=".$db;

// create PDO
try{
    $connexion = new pdo($dsn,$user,$pass);
    $connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    /* Autres Method de déclaration :
    $connexion = new PDO($dsn, $user, $pass, [ PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]); */
}
catch(PDOException $e){
    die('Errore  : ' . $e->getMessage());
}
finally{
    $connexion = null;
}
```

16. Créer une base de données et table à MySQL :

```php
// 1- Se connecter à MySQL.
$connexion = new PDO($dsn,$user,$pasw);

// 2- Création d'une base de données.
$db = "DB_1";
$res = $connexion->exec("CREATE DATABASE IF NOT EXISTS ".$db);
if($res){ echo "Création de la base de données réussie!!!!<br><br>"; }

// 3- Création d'une table.
$connexion->exec("use {$db}");
$res = $connexion->exec("CREATE TABLE IF NOT EXISTS  post(
  Id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  Nom VARCHAR(30),
  Prenom VARCHAR(30))");
if($res){ echo "Table créée avec succès!!!<br><br>"; }
```

```
Exécuter des requêtes SQL :
---------------------------
* PDO::exec :
------------
- Exécuter des requêtes SQL qui modifient les données de la base de données, ( Requêtes INSERT, UPDATE ou DELETE ).
- Elle renvoie le nombre de lignes affectées par la requête.
- Elle ne renvoie pas un objet PDOStatement pour parcourir les résultats.
```

17. Supprimer d'une base de données et table à MySQL :

```php
# Supprimer Databse
$connexion->exec("DROP DATABASE IF EXISTS ".$db);
echo "Supprimer DATABASE avec Sucess !";

# Supprimer la table
$connexion->exec("DROP TABLE IF EXISTS post");
echo "Supprimer Table avec Sucess !";
```

18. Insertion de données dans des tables :

a- PDO::exec :

```php
// 4- ( Avec Exec ).
$connexion->beginTransaction(); // // Démarrer une transaction
$nb_row = $connexion->exec("INSERT INTO post(Nom,Prenom) Values('Hamza','El khanchoufi'),('Ilyass','El khanchoufi')");
if ($nb_row) {
  echo "Nb Ligne Ajoutées : " . $nb_row."<br><br>";
  $connexion->commit(); // // Valider la transaction
 }
else {
  echo "Erreur lors de l'ajout.<br><br>";
  $connexion->rollBack(); // En cas d'erreur, annuler la transaction
}
```

19. Les requêtes MySQL préparées :

a- Avec execute (array) et des marqueurs nommés :

```php
// 5- Insérer ( Requête préparée + Paramètres nommés + execute(array) )
$stm = $connexion->prepare("INSERT INTO post(Nom,Prenom) Values(:NOM,:Prenom)");
$stm->execute([ 'NOM' => "Hamza", 'Prenom' => "El-khanchoufi" ]);
# Les clés d'un tableau associatif peuvent également être préfixées par des deux-points (":").
$stm->execute([ ':NOM' => "Hamza", ':Prenom' => "El-khanchoufi" ]);
echo "Ajouter Ligne Bien Sucess!!!!<br><br>";
```

```
Préparer la requête SQL avec des paramètres nommés :
-----------------------------------------------

Lorsque vous exécutez une requête préparée en passant les valeurs dans un tableau associatif à l'aide de execute(array),
l'ordre des paramètres dans le tableau doit correspondre à l'ordre des marqueurs de positionnement dans la requête préparée.

Préparer la requête SQL  :
-------------------------
* Prépare une requête à l'exécution et retourne un objet PDOStatement ou false en cas d'erreur.
* public PDO::prepare(string $query): PDOStatement|false
* La requête  peut contenir zéro ou plusieurs paramètres nommés (:nom) ou marqueurs (?)
* Les paramètres seront substitués par les valeurs réelles lors de l'exécution de la requête.
* Les paramètres nommés et les marqueurs ne peuvent pas être mélangés dans une seule requête préparée.
* Les paramètres nommés sont spécifiés dans la requête préparée sous la forme de :nom.
* Les paramètres marqueurs sont spécifiés dans la requête préparée sous la forme de ?.

Exécuter la requête Préparer SQL :
---------------------------------
* PDOStatement::execute — Exécute une requête préparée.
* public PDOStatement::execute(?array $params = null): bool
* Un tableau de valeurs avec autant d'éléments qu'il y a de paramètres à associer dans la requête SQL qui sera exécutée.
* Toutes les valeurs sont traitées comme des constantes PDO::PARAM_STR.
* Cette fonction retourne true en cas de succès ou false si une erreur survient.
```

b- Avec execute(array) et des marqueurs interrogatifs :

```php
// Insérer ( Requête préparée + execute(array) + marqueurs interrogatifs)
$stm = $connexion->prepare("INSERT INTO post(Nom,Prenom) Values(?,?)");
# l'ordre commence à partir de 0.
$date = array( 0=> "Hamza", 1 => "El-Khanchoufi");
$stm->execute($date);
echo "Ajouter Ligne Bien Sucess!!!!<br><br>";
```

c- En utilisant bindParam et des marqueurs nommés ou interrogatifs

```php
// Insérer  ( requêtes préparées) +  marqueurs nommés + bindParam
$stm = $connexion->prepare("INSERT INTO post(Nom,Prenom) Values(:NOM,:Prenom)");
$stm->bindParam(":NOM",$NOM,PDO::PARAM_STR);
$stm->bindParam(":Prenom",$Prenom,PDO::PARAM_STR);
$NOM="Hamza"; $Prenom="Jadore";
$stm->execute();
echo "(Marqueurs Nommés) : Ajouter Ligne Bien Sucess!!!!<br><br>";

// Insérer  ( requêtes préparées) +  marqueurs interrogatifs + bindParam
$stm = $connexion->prepare("INSERT INTO post(Nom,Prenom) Values(?,?)");
$stm->bindParam(1,$NOM,PDO::PARAM_STR);
$stm->bindParam(2,$Prenom,PDO::PARAM_STR);
$NOM="Ilyas"; $Prenom="Ahmed";
$stm->execute();
echo "(Interrogatifs) : Ajouter Ligne Bien Sucess!!!!<br><br>";

echo "La dernier ID ajouter est :". $connexion->lastInsertId()."<br><br>" ;
```

- **BindParam :**

```
BindParmatre :
-------------
* Lie une variable PHP à un marqueur nommé ou interrogatif correspondant dans une requête SQL
* la variable est liée en tant que référence et son contenu sera évalué au moment de l'appel à PDOStatement::execute().
* Pour marqueurs nommés           : ce sera le nom du paramètre sous la forme :name.
* Pour marqueurs interrogatifs : ce sera la position indexée +1 du paramètre, correspond à l'ordre des marqueurs dans la requête.
* Cette fonction retourne true en cas de succès ou false si une erreur survient.
```

- **BindValue :**

```
BindValue :
-----------
* Associe une valeur à un paramètre
* Cette méthode lie une valeur directe à un paramètre dans la requête.
* La valeur est évaluée au moment de l'appel à bindValue.
* Cette valeur liée ne changera pas même si la variable originale change après la liaison.
* Pour marqueurs nommés            : ce sera le nom du paramètre sous la forme :name.
* Pour les marqueurs interrogatifs  : ce sera la position indexée +1 du paramètre.
* Cette fonction retourne true en cas de succès ou false si une erreur survient.
```

20. Mettre à jour des données dans une table

```php
// Mettre à jour des données dans une table.
$stm = $connexion->prepare("UPDATE post set Nom=?,Prenom=? where ID>?");
$stm->bindParam(1,$NOM,PDO::PARAM_STR);
$stm->bindParam(2,$Prenom,PDO::PARAM_STR);
$stm->bindParam(3,$Prenom,PDO::PARAM_INT);
$NOM="Ilyas";  $Prenom="Ahmed";  $ID=1;
$stm->execute();
$count = $stm->rowCount();
print('Mise à jour de ' .$count. ' entrée(s) <br><br>');
```

21. Supprimer des données d'une table

```php
// Supprimer une ou plusieurs entrées choisies d'une table.
# Pour supprimer des données d'une table, nous allons utiliser l'instruction SQL DELETE FROM.
$stm = $connexion->prepare("DELETE FROM post where ID > ?");
$stm->bindParam(1,$Prenom,PDO::PARAM_INT);
$stm->execute();
$count = $stm->rowCount();
print('Effacement de ' .$count. ' entrées.<br><br>');
```

```php
// Supprimer complètement une table de la base de données
$stm = $connexion->exec("DROP TABLE post");
print('Table bien supprimée.<br><br>');
```

22. Modifier la structure d'une table

```php
// Modifier la structure d'une table
// ---------------------------------------

// Ajouter une colonne dans une table
$stm = $connexion->exec("ALTER TABLE post ADD DateInscription TIMESTAMP");
echo 'Colonne ajoutée<br><br>';

// Supprimer une colonne dans une table.
$stm = $connexion->exec("ALTER TABLE post  DROP COLUMN DateInscription");
echo 'Colonne supprimée<br><br>';

// Modifier une colonne dans une table.
$stm = $connexion->exec("ALTER TABLE post MODIFY COLUMN Prenom VARCHAR(50)");
echo 'Colonne Modifier<br><br>';
```

23. La sélection simple de données dans une base de données

A- Récupération avec la méthode query()

```
Voici les methods de récupérer des données d'un jeu de résultats :

Configurez le mode de récupération par défaut pour toutes les requêtes
----------------------------------------------------------------------
* PDO->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);

Fetch (Mode):
-------------
* une méthode de la PDOStatementclasse.
* Récupère une seule ligne d'un jeu de résultats et déplace le pointeur interne vers la ligne suivante du jeu de résultats.
* l'appel suivant à la fetch()méthode renverra la ligne suivante du jeu de résultats.
* Pour récupérer toutes les lignes d'un ensemble de résultats, vous pouvez utiliser une boucle while.
* La valeur de retour est false en cas d'échec ou s'il n'y a plus de lignes.
* FETCH_BOTH / FETCH_NUM / FETCH_ASSOC

FetchAll (Mode) :
-----------------
* méthode vous permet de récupérer toutes les lignes d'un ensemble de résultats associé à un PDOStatementobjet dans un tableau.
* Un tableau vide est retourné s'il y a zéro résultat.
* FETCH_COLUMN   :  (int $mode = PDO::FETCH_COLUMN, int $column)
* FETCH_CLASS    :  (int $mode = PDO::FETCH_CLASS, string $class)

FetchColumn(int $column = 0) :
------------------------------
*  permet de récupérer une colonne depuis la ligne suivante d'un jeu de résultats.
* La méthode prend un paramètre optionnel, $column, qui spécifie le numéro de la colonne à récupérer (en commençant à 0).
* Si aucun numéro de colonne n'est spécifié, la méthode récupérera la première colonne.
* Si aucune ligne n'est disponible, la méthode retourne false.

FetchObject($class = "stdClass") :
----------------------------------
* Récupère la prochaine ligne et la retourne en tant qu'objet.
* == fetch( PDO::FETCH_OBJ )
* Retourne une instance de la classe demandée avec les propriétés de noms qui correspondent aux noms des colonnes.
* Retourne  :  false si une erreur survient.
```

```
Modes de récupération de données avec PDO
-----------------------------------------
= Mode : Contrôle comment les données récupérées à partir d'un jeu de résultats sont renvoyées à l'appelant.
-------------------------------------------------------------------------------------------------------------
* FETCH_BOTH   : Renvoie un tableau indexé à la fois par le nom de colonne et le numéro de colonne indexé 0.
* FETCH_NUM    : Retourne un tableau indexé par le numéro de la colonne, commençant à 0
* FETCH_CLASS  : Retourne une nouvelle instance de la classe demandée, liant les colonnes du jeu de résultats aux noms des propriétés de la classe.
* FETCH_OBJ    : Retourne un objet (StdClass) avec les noms de propriétés qui correspondent aux noms des colonnes.
* FETCH_ASSOC  : Renvoie un tableau indexé par nom de colonne.
* FETCH_COLUMN : Récupérer une seule colonne spécifiée depuis la ligne suivante du jeu de résultats.
```

```
Définit le mode de récupération par défaut pour requête.
---------------------------------------------------------
* PDO PDOStatement::setFetchMode();
*   PDOStatement::setFetchMode(int $mode = PDO::FETCH_COLUMN, int $colno)
*   PDOStatement::setFetchMode(int $mode = PDO::FETCH_CLASS, string $class)
*   PDOStatement::setFetchMode(int $mode = PDO::FETCH_INTO, object $object)
```

B- Récupération avec avec une instruction préparée

C- Method des récupération des données

```php
// La sélection simple de données :
// -------------------------------

 # Obtenez la valeur de l'attribut PDO::FETCH_DEFAULT
 // FETCH_BOTH
 echo $connexion->getAttribute(PDO::ATTR_DEFAULT_FETCH_MODE);
 // Configurez le mode de récupération par défaut pour toutes les requêtes
 $connexion->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);

 #1- Récuperer les données avec la méthode query()
 $stm = $connexion->query("select * from post");
 echo "\n<pre>"; print_r( $stm->fetchAll(PDO::FETCH_OBJ) ); echo "<pre>";
```

```php
#2- 2) Fetch
$stm->execute();
while($row = $stm->fetch(PDO::FETCH_ASSOC)){
  echo "ID: {$row['Id']} | Nom: {$row['Nom']} | Prenom: {$row['Prenom']} \n";
}

$stm->execute();
while($row = $stm->fetch(PDO::FETCH_NUM)){
  echo "ID: {$row[0]} | Nom: {$row[1]} | Prenom: {$row[1]} \n";
}

$stm->execute();
while($row = $stm->fetch(PDO::FETCH_OBJ)){
  echo "ID: {$row->Id} | Nom: {$row->Nom} | Prenom: {$row->Prenom} \n";
}
```

```php
#2- 1) fetchAll
$stm->execute();
echo "\n FetchAll + mode Associative \n";
$result = $stm->fetchAll(PDO::FETCH_ASSOC);

$stm->execute();
echo "\n FetchAll + mode column \n";
$result = $stm->fetchAll(PDO::FETCH_COLUMN,1);

$stm->execute();
0 references | 0 implementations
class post{}
echo "\n FetchAll + mode Class \n";
$result = $stm->fetchAll(PDO::FETCH_CLASS,"post");
```

```php
#2- 3) FetchColumn
$stm->execute();
while($row = $stm->fetchColumn(1)){
  echo "Name:".$row."\n";
}
#2- 4) fetchObject
$stm->execute();

while($row = $stm->fetchObject("post")){
  echo "ID: {$row->Id} | Nom: {$row->Nom} | Prenom: {$row->Prenom} \n";
}
```

```
Methods et Attrbits :

Class PDO :
-----------
PDO::lastInsertId — Retourne l'identifiant de la dernière ligne insérée ou la valeur d'une séquence
PDO::setAttribute — Configure un attribut PDO


Class PDOStatement :
--------------------
PDOStatement::errorCode — Récupère les informations sur l'erreur associée lors de la dernière opération sur la requête
PDOStatement::errorInfo — Récupère les informations sur l'erreur associée lors de la dernière opération sur la requête
PDOStatement::rowCount — Retourne le nombre de lignes affectées par le dernier appel à la fonction PDOStatement::execute()  ( DELETE, INSERT ou UPDATE )
PDOStatement::setAttribute — Définit un attribut de requête
PDOStatement::setFetchMode — Définit le mode de récupération par défaut pour cette requête
```