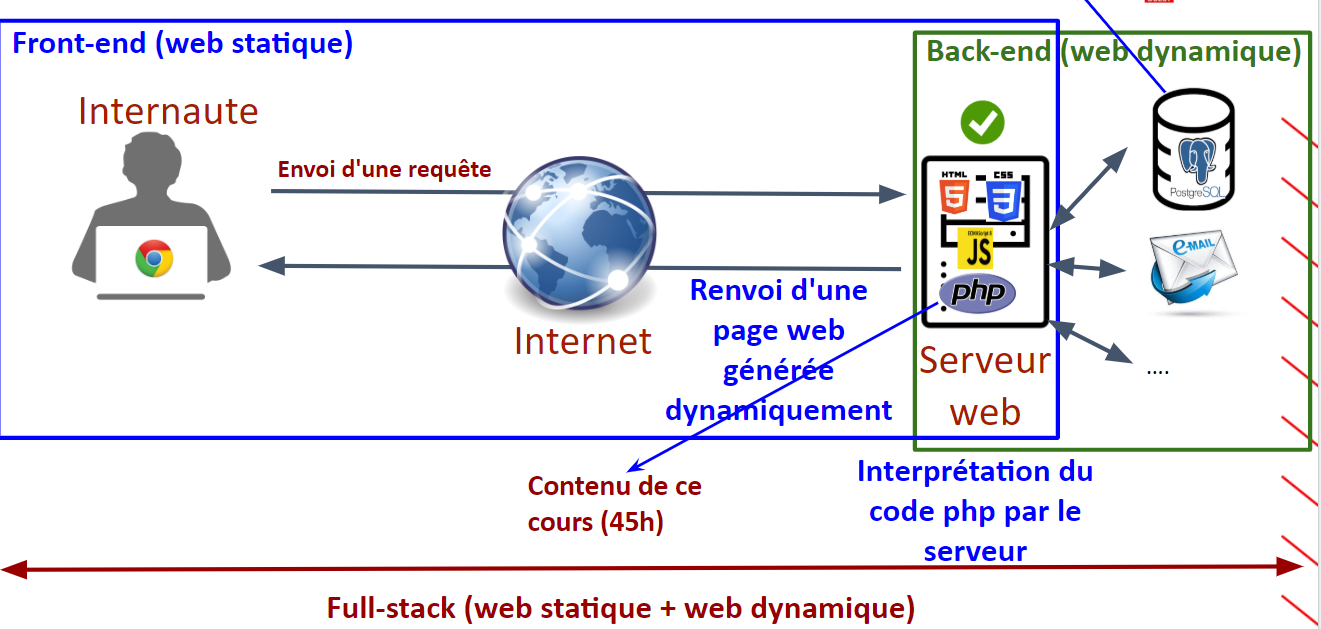
|  |
| --- |
| COUR PHP INTRODUCTION |



PHP (Hypertext Preprocessor) est un langage de programmation interprété qui s’exécute côté serveur pour donner du dynamisme à une page HTML.

* Un code PHP s'écrit entre les balises <?php (ouverture) et ?> (fermeture)
* Chaque instruction PHP est suivie de ";"

Commentaires en PHP (pareil que les langages C et C++) :

On peut écrire directemenet le code php dans un page html ou sinon entre des balises < ?php > on include notre fichier php avec le code

A l’inverse on peut afficher de l’html en php avec la commande :

Echo «<h1>title</h1> » ;

On a aussi besoin :

1. Installation d'un serveur web http (apache2 dans notre cas )
2. Installation du module PHP (Zend Engine est automatiquement installé comme interpréteur) dans le serveur apache avec le terminal WSL par exemple (sinon le code n’est pas interprété )

|  |
| --- |
| COUR PHP VARIABLES & CONSTANTES |

Nom des variables :

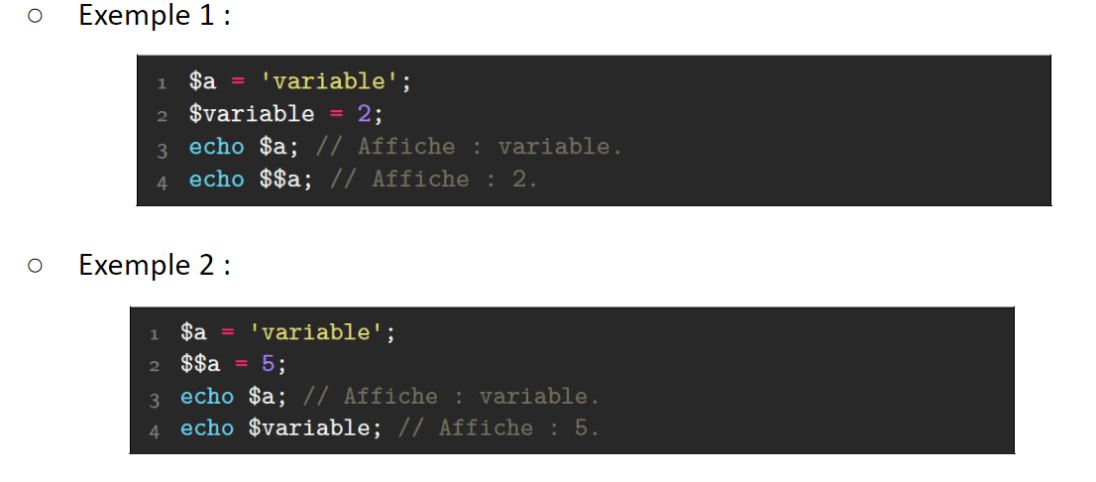
Avant le nom de la variable il y a un ‘$’ le nom de la variable est composé de chiffre lettre underscore et est sensible à la casse

Le type de la variable est détecté automatiquement

Array un tableau association de valeurs et de clé -> un tableau une liste un table de hashage un dictionnaire, une collection, une pile, une file d’attente …

Chaine de caractère :

Double guillemets : dans ce cas, PHP interpole les variables et les caractères d’échappement à l’intérieur de la chaîne de caractères.    
Simple guillemets : **(apostrophes)** : aucune interpolation de variable ou de caractère d’échappement n’est possible;

Opération de chaine de caractère :

Concaténation ex)

Echo ‘Hello ‘. ‘World ‘ ;

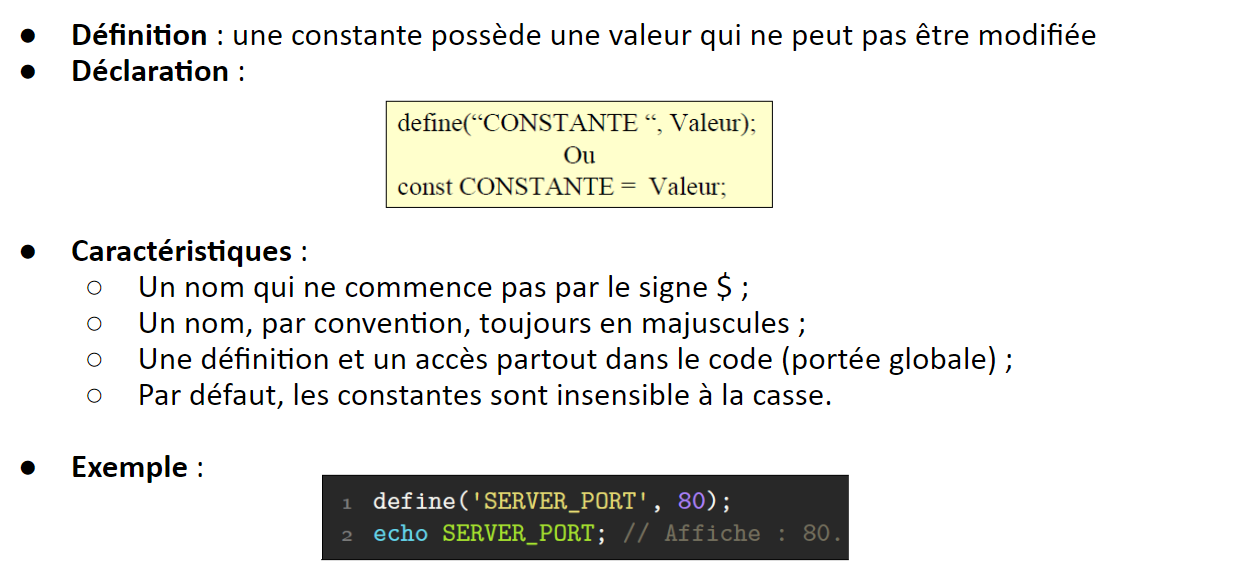
Ou

&var = ‘Hello ‘ ;

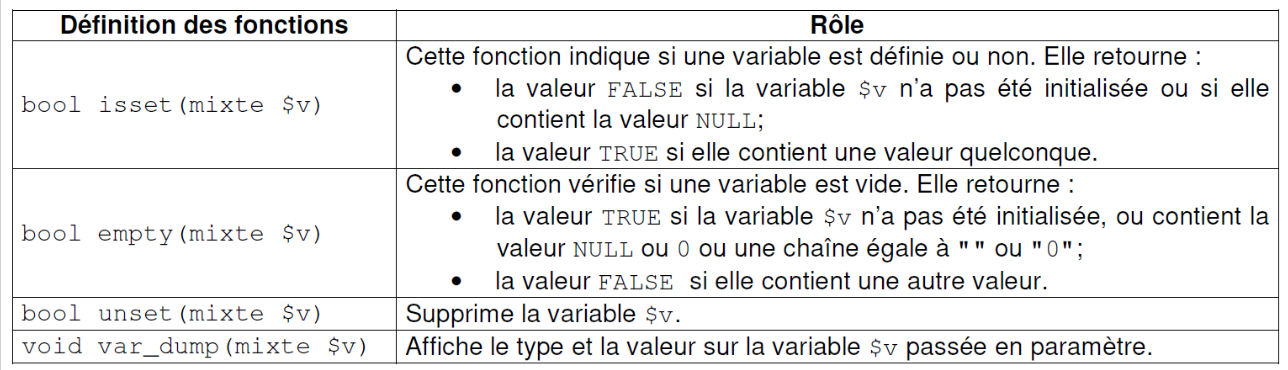
&var .= ‘World ‘ ;

Echo « $var » ;

Constantes :



Controle de l’etat d’une variable :

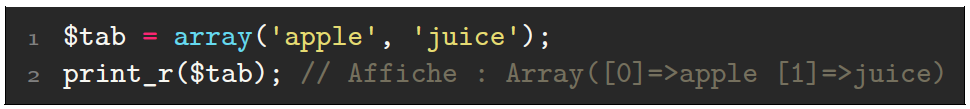
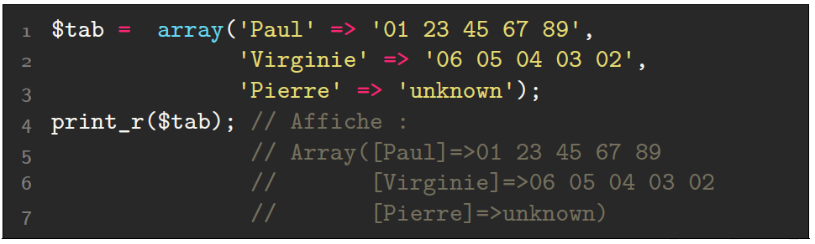
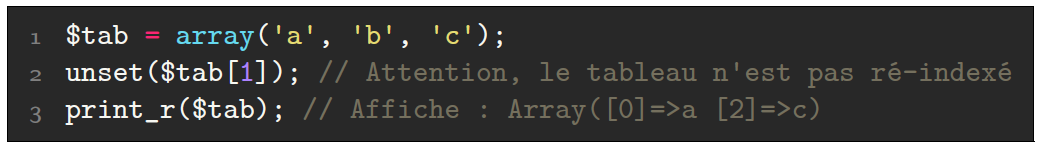
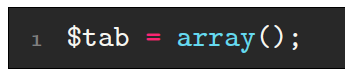


|  |
| --- |
| COUR PHP – Opérateurs, tableaux et tuctures de controle |

Opérateurs :

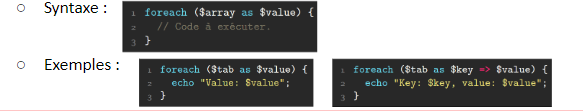
* Les opérateurs arithmétiques :
* Les opérateurs d’affectation :
* Les opérateurs de comparaison :
* Les opérateurs d’incrémentation / décrémentation :
* Les opérateurs logiques :
* Les opérateurs sur les chaînes de caractères (voir CM2)

Tableaux :

* Le tableau PHP regroupe en un seul type, array, les deux notions suivantes :
  + **Les tableaux indicés** : indicé par des entiers positifs (tpe de la clé : entier) ;
    - Déclaration
  + **Les tableaux associatifs** : indicé par des chaînes de caractères (type de la clé : chaîne)
    - Déclaration
* Suppression d'un élément (utilisation de la fonction unset) :  
  
* Suppression de tous les éléments :  
  
* Suppression d'un tableau (utilisation de la fonction unset):



Boucle Foreach :



|  |
| --- |
| COUR PHP – Fonctions |

# Fonctions :

Ex

Function ‘nom\_function’(arg1, arg2, …){

Instructions1 ;

Instructions2 ;

}

* La portée des variables à l’intérieure d’une fonction modifie la visibilité de cette variable dans l’ensemble du code :
  + Locale (à la fonction) : visible uniquement à l'intérieur de la fonction
  + static : persiste (dans la fonction) durant la durée du script appelant
  + global : visible dans et hors de la fonction

# Fonction d’affichage :

* echo : s’utilise avec ou sans parenthèses et affiche ses paramètres ;
* print : identique à echo, sauf qu’un seul paramètre est permis ;
* printf : fonctionne comme la fonction C de même nom ;
* print\_r : affiche la valeur d’une variable ;
* var\_dump : affiche des informations complètes à propos d’une variable.

# Fonctions sur chaine de cara :

* strlen($string) : renvoie la longueur de $string ;
* str\_word\_count($string) : renvoie le nombre de mots dans $string ;
* strrev($string) : renvoie la chaîne inverse de $string ;
* strpos($string, $substring) : renvoie la position de $substring dans $string (false si non trouvé) ;
* str\_replace($word1, $word2, $string) : renvoie la chaîne $string où $word1 à été remplacé par $word2 ;
* substr($string, $pos, $len) : renvoie la sous-chaîne démarrant à la position $pos et de longueur $len ;
* strtolower($string) : renvoie la chaîne $string en minuscules ;
* strtoupper($string) : renvoie la chaîne $string en majuscules ;
* intval($string) : renvoie le nombre correspondant à la chaîne $string.
* intval(string $haystack , mixed $needle) : trouve la première occurrence dans une chaîne

# Fonction sur les tableaux :

* **Fonctions de tri** :
  + sort : tri, selon les valeurs, un tableau indicé par des entiers ;
  + asort : tri, selon les valeurs, un tableau indicé par des clés ;
  + ksort : tri, selon les clés, un tableau indicé par des clés.
* **Fonctions de modification** :
  + array\_shift : retourne le premier élément d’un tableau et le supprime
  + array\_pop : retourne le dernier élément d’un tableau et le supprime ;
  + array\_push : ajoute un (des) élément(s) à la fin d’un tableau.
* **Fonctions sur les clés et les valeurs** :
  + array\_keys : renvoie sous forme de tableau l’ensemble des clés ;
  + array\_values : renvoie sous forme de tableau l’ensemble des valeurs ;
  + array\_key\_exists : renvoie VRAI si une clé existe ;
  + in\_array : renvoie VRAI si une valeur existe.

|  |
| --- |
| COUR PHP – Dates |

# Fonction liées aux dates et aux heures :

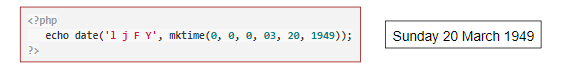
Time (void) : int

Renvoie le timestamp unix (nombre de secondes ecoulées depuis le 1er janvier 1970 ah 00h00)

Mktime(int $hour, int $minutes, $second, $month, $day, $year): int

Renvoie le timestamp d’une date fournie en paramètre

Date(string $format, $timestamp)



* **Rôle** : Représentation d'une date et heure
* **Constructeur** :  DateTime::\_\_construct (string $time = "now" , DateTimeZone $timezone = NULL)
  + **$time** : une chaîne date/heure
  + **$timezone** : un objet de la classe DateTimeZone pour représenter le fuseau horaire

Changement du format via la méthode DateTime::format ( string $format ) : string

**Solution** : utilisation de

* setlocale ( int $category , string $locale) : string  ⇒ modification des informations de localisation

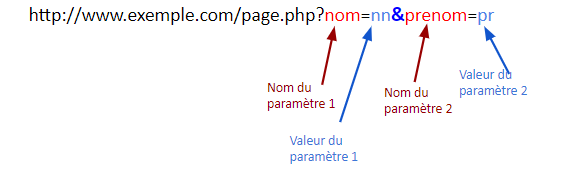
strftime ( string $format, int $timestamp) : string ⇒ formatage de la date/heure avec la configuration locale

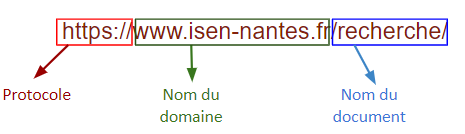
* Récupération du timestamp Unix via la méthode DateTime::getTimestamp ( void ) : int
* Ajout d'une durée via la méthode DateTime::add(DateInterval $interval)
* Suppression d'une durée via la méthode DateTime::sub(DateInterval $interval)

Calcul de la différence entre deux dates DateTime::diff(DateTimeInterface $datetime2)

|  |
| --- |
| COUR PHP – Transmission des données |

# URL

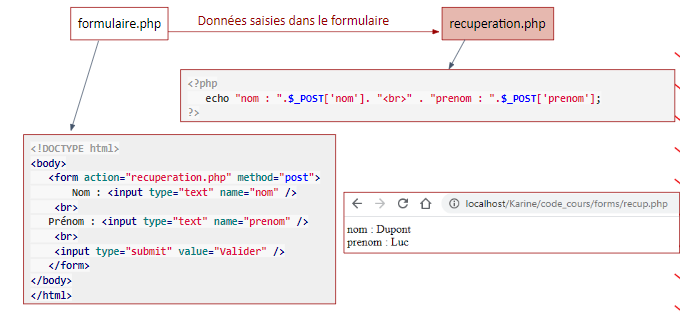
* Contenu basique d'une URL :  le protocole (comment), le nom du domaine (où) et le nom du document (quoi)



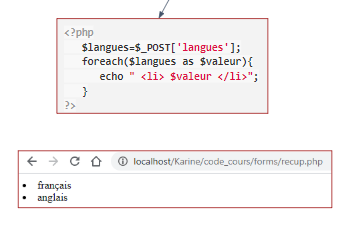


# Formulaires

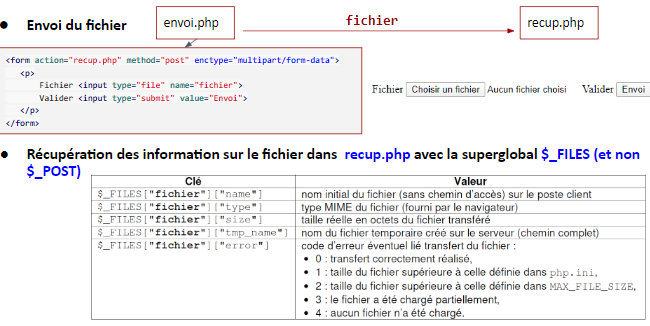
* La plupart des échanges entre un visiteur et un serveur passe par un formulaire
* Les valeurs entrées dans un formulaire sont soit uniques soit multiples :
  + **Valeurs uniques** : l'utilisateur ne peut entrer qu'une seule valeur
  + **Valeurs multiples** : l'utilisateur peut entrer plusieurs valeurs





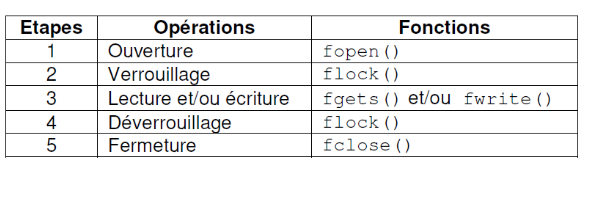


Transfert d’un fichier vers le serveur :

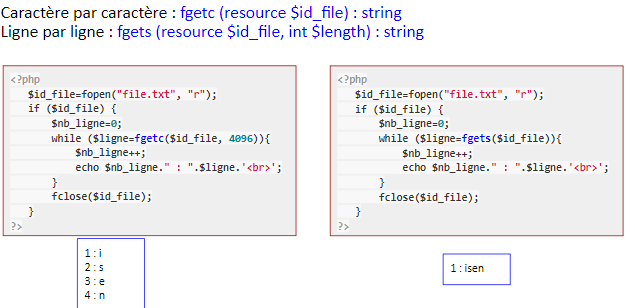
* Enregistrement du fichier téléchargé dans le serveur :   
  boolean move\_uploaded\_file(string $fichier\_tmp, string $fichier\_final)
  + $fichier\_tmp : chemin d’accès au fichier temporaire indiqué dans $\_FILES (valeur de la clé tmp\_name)
  + $fichier\_final : nom sous lequel est enregistré définitivement le fichier sur le serveur (par défaut, dans le même répertoire que le script en cours d'exécution).



# Cookies & Session

* Écriture des cookies : setcookie (...)
* Lecture et affichage d'un cookie : echo $\_COOKIE['nomcookie']
* Affichage de tous les cookies :  print\_r($\_COOKIE)
* Démarrage d'une session : session\_start()
  + doit être en début du script
  + doit être dans tous les scripts (appelant)
* Un identifiant (numéro de session) est généré pour le visiteur et lui ai envoyé via un cookie de nom PHPSESSID. Pour le récupérer, on utilise session\_id()
* Définition d'une session : $\_SESSION['mavar']= mavaleur
* Destruction d'une session : session\_destroy()

# Fichiers



|  |
| --- |
| COUR PHP – Interaction avec les bases de données |

**Approche procédurale**

mysqli

pgsql

…

**Approche orientée objet**

PDO (PHP Data Objects) à partir de PHP5

...

PDO utilise 3 classes principales :

PDO : qui gère la connexion à la BDD et exécute les requêtes

PDOStatement : qui correspond aux requêtes et aux résultats (des requêtes)

PDOException : qui correspond aux erreurs de PDO

# Connexion à la BDD

|  |  |
| --- | --- |
| Exemple avec MySQL | **<?php** $dsn = 'mysql:dbname=testdb;host=127.0.0.1'; $user = 'dbuser'; $password = 'dbpass';  **try** {     $conn = **new** PDO($dsn, $user, $password); } **catch** (PDOException $e) {     **echo** 'Connexion échouée : ' . $e->getMessage(); } **?>** |
| Exemple avec Postgre | **<?php** $dsn = 'pgsql:dbname=testdb;host=127.0.0.1;port=5432'; $user = 'dbuser'; $password = 'dbpass';  **try** {     $conn = **new** PDO($dsn, $user, $password); } **catch** (PDOException $e) {     **echo** 'Connexion échouée : ' . $e->getMessage(); } **?>** |

# Accès en lecture :

|  |  |
| --- | --- |
| Query (pour les requêtes sans paramètres ) | <?php      $query1 = "SELECT \* from voyage";      $result1 = $conn->query($query1);      foreach($result1 as $data)      {          echo $data['id']."\n";      }  ?> |
| Prepare (pour les requêtes avec paramètre) | $query = "SELECT libelle,date FROM voyage where id=1"  $sth = $conn->prepare($query); $sth->execute(); |
| Execute (pour les requêtes avec paramètre ) |

|  |  |
| --- | --- |
| Fetch pour les requêtes avec paramètre -> récupère la ligne suivante d’une table | $query = "SELECT libelle,date FROM voyage where id=1"  $sth = $conn->prepare($query);  $sth->execute();  $result=$sth->fetch(); |
| FetchAll pour les requêtes avec paramètre retourne un tableau contenant toutes les lignes de la table | $query = "SELECT libelle,date FROM voyage where id=1"  $sth = $conn->prepare($query);  $sth->execute();  $result=$sth->fetchAll(); |

# Accès en écriture

|  |  |
| --- | --- |
| Insertion en utilisant une requête non préparée | **<?php** $sql = "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('John', 'Doe', 'john@example.com')";  **if** ($conn->query($sql) === **TRUE**) {     **echo** "New record created successfully"; } **else** {     **echo** "Error: " . $sql . "<br>" . $conn->error; } **?>** |
| Suppression en utilisant une requête non préparée | $conn->exec("DELETE FROM fruit"); |

|  |  |
| --- | --- |
| Insertion en utilisant une requête préparée | **<?php** $sql = "INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)"; $sqlR = $conn->prepare($sql); $sqlR->execute([$var1, $var2, $var3]) **?>** |
| Suppression en utilisant une requête préparée | **<?php** $sql = "DELETE from MyGuests WHERE id=?"; $sqlR = $conn->prepare($sql); $sqlR->execute([$var1]) **?>** |