# DataBase(php)

## Question:

### Redirection:

Pour la redirection ou doit être placé le header car il est dit avant toute balise html c'est à dire en tout tout début de code ???

#### Réponse: (en cour)

Il est obligatoire de placer une redirection en première ligne

### Destruction variables:

Est ce vraiment une destruction il a été dit en cours que la variable était supprimé et plus tard que la variable prenais la valeur : **null**
et avec les fonctions de vérification d'existence d'une variable ne différencie pas entre existante et null mais est ce qu'un garbage collector supprime les variables avec null ou reste telle inutilisé et ne prenne pas de mémoire.

#### Réponse: (en cour)

## Inclusion redirection:

### Les 4 instructions:

- **include**: Inclus le fichier.
- **include_once**: N'inclus le fichier que si il ne l'est pas encore.
- **require**: Si fichier introuvable donne un message d'erreur.
- **require_once:** N'inclus le fichier que si il ne l'est pas encore

## Rediriger:

On utilise une fonction qui en brut envoie l'adresse http au serveur.
Le exit mets fin à l'interprétation du code ( il est recommandé de l'utiliser comme bonne pratique)

**Ex:**
header(*'url ou l'on souhaite rediriger l'utilisateur')*
exit

# Variable:

## Règle pour les noms de variables:

- Commence par l'identifiant $ (pense argent)
- Ne peut comporter que des lettres (pas les accents zebi ta le droit mais vaut mieux pas), des chiffres et le fameux _ .
- Pas de chiffre directement après le $
- Sensible à la casse (si il y a des majuscules c'est **IMPORTANT**)
- Pas de limite de longueur ( Malaise a dit il y a une limite juste elle est grande)

## Affectations, déclarations et destructions:

### Affectations et déclarations :

Les variables n'ont pas de type et peuvent changer de type comme **python** une variable se déclare comme en **python** un exemple vaut mieux que mille mots.

**Ex:**

```
<?php
        $x = 10;
?>
```

### Destructions:

Pour détruire on utilise la fonction :

***unset(*** *$NomDeLaVariable* ***);***

## Tester l'existence d'une variable:

| | La variable est inexistante | La variable est de type NULL | La variable vaut 0 ou '' | La variable a une valeur différente de 0 |
|---|---|---|---|---|
| `isset($var)` | **False** | **False** | **True** | **True** |
| `empty($var)` | **True** | **True** | **True** | **False** |
| `!isset($var)` | **True** | **True** | **False** | **False** |
| `!empty($var)` | **False** | **False** | **False** | **True** |

## Affectation par référence:

**1. Affectation par valeur :**

```php
<?php
$A = 'Mons';
$B = 'Bruxelles';
$A = $B;
$B = 'Charleroi';
?>
```

A la fin de ce script, $A vaut "Bruxelles"

**2. Affectation par référence (&) :**

```php
<?php
$A = 'Mons';
$B = 'Bruxelles';
$A = &$B;
$B = 'Charleroi';
?>
```

A la fin de ce script, $A vaut "Charleroi"

### Affectation par valeur:

Dans le premier cas ( $A = $B; )on dit que la **valeur** de $A vaut la **valeur** de $B donc si la valeur de $B change après l'affectation de la valeur $B → $A, $A ne change pas.

### Affectation par l'adresse (référence &):

Dans le deuxième cas on écrit ( $A = &$B ) ce qui veut dire que $A devient un pointeur($A ne vaut pas la valeur de $B mais son adresse mémoire), donc quand on affichera $A si l'on a modifié $B alors $A aura la nouvelle valeur de $B (on peut relier ça à un alias en python quand l'on copiait des variables).

## Constante:

Si valeur qui va se répéter et reste inchangé il faut utiliser une constante

```php
<?php
const WIDTH = 800 ;
define ("PI", 3.1415926535, TRUE);
?>
```

- Pas besoin de $
- Define permet de rendre sensible ou non à la casse (**TRUE** = insensible (*tant qu'on écrit pi ça passe*), **FALSE** = sensible (*faut écrire à l'identique*)

## Types de variables:

- entier : **integer** (-4 ou 36)
- réel (ou flottant) : **double** (51.8)
- booléen : **boolean** (True ou False)
- chaîne de caractères : **string** ("Martine" ou 'Guy')
- tableau: **array**
- objet : **object**
- ressource : **resource**
- néant : **NULL**

**gettype();**

permet d'obtenir un string d'une des valeurs en gras ci- dessus.

```php
is_integer($x)      is_double($x)      is_array($x)
is_int($x)          is_numeric($x)     is_object($x)
is_float($x)        is_bool($x)        is_resource($x)
is_real($x)         is_string($x)      is_null($x)
```

Retourne un booléen en fonction du type de la variable.

```php
$x = true;   //$x est un booléen
$x = 23;     //$x devient un entier
$x = 0.0;    //$x devient un réel
$x = 'Bob'; //$x devient une chaîne de caractères
```

**OU**

```php
$var = '7000 Mons';    //$var est de type string
$var = (int) $var;     //$var devient integer et vaut 7000
$var = (bool) $var;    //$var devient bool et vaut True
```

## Chaine de caractère:

le cours l'explique bien

1. Les guillemets causent des collisions avec les guillemets ; les apostrophes avec les apostrophes. Pour éviter les collisions, il faudra échapper les caractères avec \ (*backslash*) :

```php
<?php
$nom1 = 'Je m\'appelle Michael "Monty" Widenius';
$nom2 = "Je m'appelle Michael \"Monty\" Widenius";
?>
```

2. les guillemets permettent l'évaluation d'une variable mais pas les apostrophes :

```php
<?php
$place = 'Mons';
$a = "J'habite $place ";  //vaut : J'habite Mons
$a = 'J\'habite $place';  //vaut : J'habite $place
```

**Caractères spéciaux accessibles avec un *backslash***

| | |
|---|---|
| \\ | Affiche un backslash (antislash) |
| \' | Affiche une apostrophe |
| \" | Affiche des guilemets |
| \$ | Affiche un $ |
| \n | Nouvelle ligne |
| \r | Retour chariot |
| \t | Tabulation horizontale |
| \115 | Nombre octal affichant le caractère correspondant : M |
| \x4D | Nombre hexadécimal affichant le caractère correspondant : M |

Remarque : si la chaîne est entre apostrophe, seuls les 2 premiers codes \\ et \' fonctionnent.

PG 17 pour plus d'info mais

**iconv_strlen()** renvoie le nombre de caractères.

**strto -lower** renvoie la chaîne avec tous les caractères en minuscule.

**-upper "** **"** en majuscule.

## Echo:

La commande permet de générer un contenu (html, CSS, javascript, texte, etc)

texte:

**echo** 'Texte simple';

**echo** “Texte simple”;

Code html:

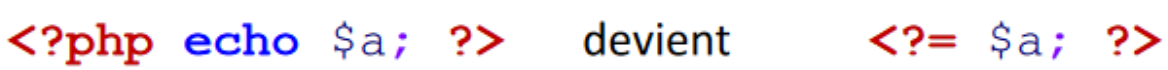
**echo** ‘<h1>Code html</h1>’;

Valeur d’une variable:

**echo** $a;
**echo** “$a”;

Nom d’une variable:

**echo** ‘$a’;




## Tableaux:

Les tableaux sont dynamiques, leur taille peut changer pendant l’exécution du script cela a pour impact qu’il peuvent prendre 20x plus de place que des tableaux statiques.

Le type des variables dans le tableau peut être différent.

Pour afficher un tableau on peut utiliser:

**var_dump(**$Tableau**);** Plus précis et donne plus d’information (type de la variable indice et valeur de la variable..
**print_r(**$Tableau**)**; Moins précis donne les éléments et leur indice.

Il existe 2 types de tableau:

- Les tableaux indicés, on retrouve l'élément grâce à l’indice (nombre) qui lui à été attribué.
- Les tableaux associatifs, on retrouve l’élément grâce au string (chaîne de caractères) qui lui à été attribué.

Fonctions sur les tableaux en page 25 du syllabus.