



W1 - Piscine PHP

W-WEB-024

Jour 05

Variables superglobales





Jour 05

repository name: poolphpday05
repository rights: ramassage-tek
language: PHP



- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

INFORMATIONS

AVANT DE COMMENCER

- Lisez attentivement toutes les consignes.
- Consultez vos mails plusieurs fois par jour, tous les jours.



Commencez par lire vos mails tout de suite à l'adresse : mail.office365.com.

- C'est une pangolinette (un programme) qui corrige vos exercices. Vérifiez le travail que vous allez rendre afin qu'il respecte scrupuleusement les consignes.
- Vous devez respecter les restrictions qui sont imposées dans chaque exercice. Le cas contraire, la pangolinette va considérer comme **triche** en attribuant la note de **-42**.
- Vous êtes susceptibles à tout moment de recevoir des corrections intermédiaires



Pour bénéficier de corrections intermédiaires, vous devez chaque jour :

- Etre inscrit au projet et aux activités dans l'intranet.
- Tenir à jour régulièrement le dépôt.

- Ne laissez jamais votre session ouverte sans surveillance.

JOUR 05



Votre répertoire ne doit pas contenir de fichiers inutiles (fichiers temporaires, ...)
N'oubliez pas de push régulièrement vos fichiers, sans cela, pas de correction



Pensez à créer votre répertoire en début de journée et à envoyer votre travail via **git**!
Le nom du répertoire est spécifié dans les instructions pour chaque étape/exercice.
Pour garder votre répertoire propre, regardez du côté de `gitignore`.



N'oubliez pas de vous inscrire à toutes les activités possibles de la semaine.



ÉTAPE 1 - (1 POINT)

Nom de rendu : ex_O1.php

Restrictions : Aucune

Créez une fonction nommée “declare_globals” qui crée les variables globales suivantes :

- a = “hello”
- b = “world
- c = “le”
- d = “monde”
- e = “n’est”
- f = “que”
- g = “PHP”
- h = “!”

Prototype : void declare_globals(void);

ÉTAPE 2 - (1 POINT)

Nom de rendu : ex_O2.php

Restrictions : Aucune

Créez une fonction “my_aff_global” qui affiche toutes les variables globales seulement si elles sont de type string. Le format sera : “[nom] => [valeur]\n”.

Prototype : void my_aff_global(void);



globale



ÉTAPE 3 - (1 POINT)

Nom de rendu : ex_O3.php

Restrictions : Aucune

Créez une fonction qui met dans un cookie la clé et la valeur données en paramètre. La valeur devra être modifiée pour rajouter **"toto"** à la fin.

Prototype : void my_add_to_cookie(string \$key, string \$value);

Exemple :

```
Terminal
my_add_to_cookie ("pseudo", "Max_");
// Cree un cookie a la clef "pseudo" et qui a pour contenu "Max_toto".
```



cookies

ÉTAPE 4 - (1 POINT)

Nom de rendu : ex_O4.php

Restrictions : Aucune

Écrire une fonction qui affiche le contenu du cookie possédant la clef définie par le premier paramètre, suivi de la chaîne de caractères **"_END"** puis d'un retour à la ligne. Si le cookie n'existe pas, rien n'est affiché.

Prototype : void my_print_cookie(string \$key);

ÉTAPE 5 - (1 POINT)

Nom de rendu : ex_O5.php

Restrictions : Aucune

Créez une fonction qui met dans une session la clé et la valeur données en paramètre. La valeur devra être modifiée pour rajouter "titi" à la fin.

Prototype : void my_add_to_session(string \$key, string \$value);

Exemple :

```
Terminal
my_add_to_session("pseudo", "Max_");
// Cree une variable de session a la clef "pseudo" ayant pour contenu "Max_titi"
```



session

ÉTAPE 6 - (1 POINT)

Nom de rendu : ex_O6.php

Restrictions : Aucune

Écrire une fonction qui affiche le contenu de la session possédant la clef définie par le premier paramètre, suivi d'un double retour à la ligne. Si la session n'existe pas, rien n'est affiché.

Prototype : void my_print_session(string \$key);

ÉTAPE 7 - (1 POINT)

Nom de rendu : ex_07.php

Restrictions : Aucune

Créez une fonction qui vide la session, la détruit et la réinitialise.

Prototype : void my_reset_session(void);

ÉTAPE 8 - (1 POINT)

Nom de rendu : ex_08.php

Restrictions : Aucune

Créez une fonction qui retourne la valeur de l'élément de **POST** possédant la clé "phelix" s'il existe ou **NULL** en cas d'erreur.

Prototype : mixed helix_post_finder(void);



POST

ÉTAPE 9 - (1 POINT)

Nom de rendu : ex_09.php

Restrictions : Aucune

Créez une fonction qui renvoie la distance de levenshtein entre l'élément "**str_one**" et "**str_two**" de **POST**. En cas d'erreur la fonction retourne **NULL**.



Prototype : mixed post_levenshtein_score(void);

Exemple :

```
Terminal
Si on a dans POST :
{
  "str_one" => "Sophie",
  "str_two" => "Julien"
}
post_levenshtein_score() renvoie : 5
```

ÉTAPE 10 - (1 POINT)

Nom de rendu : ex_10.php

Restrictions : Aucune

Créez une fonction qui affiche un certain nombre d'éléments du **GET** et renvoie **0**. Ce nombre est lui-même passé en paramètre à **GET** avec la clef "**nbr_elem**". En cas d'erreur, retourne **NULL**.

Prototype : mixed my_get_weird_info(void);

Exemple :

```
Terminal
Si mon GET contient :
$_GET["nbr_elem"] = 5
$_GET["var1"] = "premier test"
$_GET["var2"] = "deuxieme test"
$_GET["var3"] = "dernier test"
$_GET["directrice"] = "Sophie"
$_GET["responsable"] = "Julien"
$_GET["pangolin1"] = "Samy"
$_GET["pangolin2"] = "Henri"
my_get_weird_info() affiche :
$_GET["nbr_elem"] = 5
$_GET["var1"] = "premier test"
$_GET["var2"] = "deuxieme test"
$_GET["var3"] = "dernier test"
$_GET["directrice"] = "Sophie"
```



GET

ÉTAPE 11 - (1 POINT)

Nom de rendu : ex_11.php

Restrictions : Aucune

Écrire une fonction qui affiche toutes les clés et valeurs du tableau passé en paramètre sous la forme “**valeur : clé**”. Un retour à la ligne sera effectué après chaque affichage.

Prototype : void print_array_with_key(array \$my_array);

ÉTAPE 12 - (1 POINT)

Nom de rendu : ex_12.php

Restrictions : Aucune

Créez une fonction utilisant la superglobale `SERVER` pour déterminer avec précision le temps qu’a pris votre script php à s’exécuter. Elle retourne ce temps en secondes avec une précision de 7 chiffres après la virgule.

Prototype : float get_execution_time(void);

Exemple :

```
Terminal
Si le script a mis 0.0021234342s a s'exécuter, la fonction retourne :
0.0021234
Si le script a mis 0.0013382s a s'exécuter, la fonction retourne :
0.0013382
Si le script a mis 0.00091923s a s'exécuter, la fonction retourne :
0.0009192
```



Serveur

ÉTAPE 13 - (1 POINT)

Nom de rendu : ex_13.php

Restrictions : Aucune

Créez une fonction **“my_sort_files”** qui prend en paramètre une référence d'un tableau de chaînes de caractères, puis qui trie les valeurs par ordre alphabétique croissant. Elle doit ensuite afficher chaque valeur du tableau suivi d'un retour à la ligne.

Prototype : void my_sort_files(array &\$tab);



“PANGOLIN” < “pangolin”



sort

ÉTAPE 14 - (1 POINT)

Nom de rendu : ex_14.php

Restrictions : Aucune

Créez une fonction **“display_names”** qui retourne un tableau contenant :

- le nom du script courant, à l'index **“0”** (Il n'est pas attendu un chemin, mais bien seulement un nom).
- le nombre d'arguments passés en paramètres à la fonction, à l'index **“3”**.
- dans l'index **“4”**, 1 si jamais le nombre d'arguments passés en paramètres à la fonction est pair sinon 0.

Prototype : `array display_names(...$params);`

ÉTAPE 15 - (1 POINT)

Nom de rendu : ex_15.php

Restrictions : Aucune

Créer une fonction **“convert_number”** qui affiche la valeur ASCII du nombre donné en paramètre. Si le nombre donné en paramètre est **supérieur à 1000**, la fonction affichera la chaîne de caractères : **“Vive les Pangolins”** à la place. Tout affichage se termine par un retour à la ligne.

Prototype : `void convert_number(int $nbr);`

ÉTAPE 16 - (1 POINT)

Nom de rendu : ex_16.php

Restrictions : Aucune

Écrire une fonction **“check_types”** qui prend un tableau à deux dimensions en paramètre et qui vérifie la correspondance entre les clés et les tableaux de valeurs associés. La fonction retourne **TRUE** si toutes les valeurs rencontrées correspondent aux bons types, sinon **FALSE**.



Prototypes : void check_types(array \$types);

Exemple :

```
Terminal
$types = ["integer" => [5121 , 454, -5464],
"double" => [22.0, 262.65, -54.54], "NULL" => [NULL, NULL, NULL],
"string" => ["a", "a", "ab"], "object" => [new stdClass()]];
check_types($types) ; // return true $types = ["array" => [[]]];
check_types($types) ; // return true $types = ["float" => [2.0]];
check_types($types) ; // return false
```

ÉTAPE 17 - (1 POINT)

Nom de rendu : ex_17.php

Restrictions : Aucune

Créez une fonction **“check_url”** qui devra vérifier que les paramètres envoyés au script via la méthode HTTP **“GET”** contiennent une clé **“token”**.

La valeur associée à la clé **“token”** doit être une chaîne dont chaque caractère doit appartenir à la **“whiteList”** passée en paramètre de la fonction (sous forme de tableau). Si la valeur du **“token”** est valide, la fonction retournera 1, sinon elle devra retourner -1.

Attention à bien vérifier l'existence de la clé **“token”** parmi les données GET, si elle n'existe pas la fonction doit retourner false.

Prototype : mixed check_url(array \$whiteList);

Exemple :

```
Terminal
// avec comme valeur de token : acbeacddf
$whiteList = array("a", "b", "c", "d", "e", "f");
check_url ($whiteList) ; // return 1
// avec comme valeur de token : 4ea3fcd1
$whiteList = array("a", "b", "c", "d", "e", "f");
check_url ($whiteList) ; // return -1
```



ÉTAPE 18 - (1 POINT)

Nom de rendu : ex_18.php

Restrictions : Aucune

Créez une fonction “**check_user_acl**” qui vérifie les permissions d'un utilisateur donné.

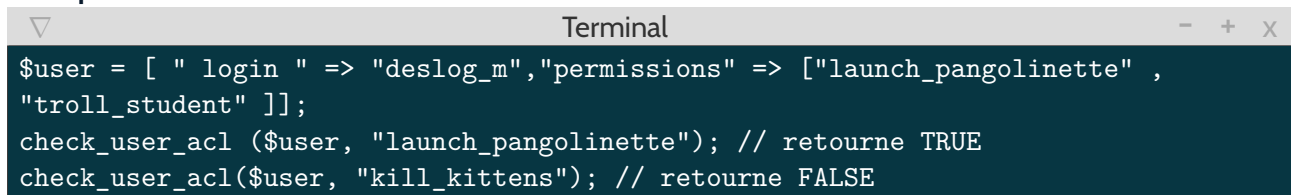
Votre fonction devra prendre 2 paramètres :

- un tableau contenant les informations d'un utilisateur
- la permission dont il faut vérifier la présence

La fonction doit retourner TRUE si l'utilisateur possède la permission, sinon FALSE.

Prototypes : bool check_user_acl(array \$user, string \$permission);

Exemple :



```
Terminal
$user = [ " login " => "deslog_m", "permissions" => ["launch_pangolinette" ,
"troll_student" ] ];
check_user_acl ($user, "launch_pangolinette"); // retourne TRUE
check_user_acl($user, "kill_kittens"); // retourne FALSE
```

ÉTAPE 19 - (1 POINT)

Nom de rendu : ex_19.php

Restrictions : Aucune

Créez une fonction “**get_authorized_users**” qui affiche le ou les logins suivis chacun d'un saut de ligne dont les utilisateurs ont les permissions demandées.

La fonction accepte deux paramètres :

- la permission dont il faut vérifier l'existence
- un tableau d'utilisateurs

Prototype : void get_authorized_users (string \$permission, array \$users);

Exemple :

```
Terminal
$users = [
  ["login" => "deslog_m",
  "permissions" => ["launch_pangolinette" , "troll_student"]],
  ["login" => "pellet_f",
  "permissions" => ["launch_pangolinette" , "troll_student"]],
  ["login" => "student_x",
  "permissions" => ["offer_sweets_to_pangolin" ]]];
get_authorized_users("launch_pangolinette", $users); // deslog_mnpellet_f\n
get_authorized_users("offer_sweets_to_pangolin", $users); // student_x\n
```

ÉTAPE 20 - (1 POINT)

Nom de rendu : ex_20.php

Restrictions : Aucune

Créez un script qui affiche **"courriel valide\n"** si les données envoyées au script contiennent un email correctement formaté, sinon la fonction affichera **"courriel invalide\n"**. Les données sont envoyées au script par un formulaire via la méthode **POST**, la valeur à vérifier se trouvera dans le champ **"email"**.

L'email sera considéré valide (correctement formaté) s'il répond à ces règles :

- ne comporte ni espace, ni tabulation
- comporte exactement un caractère **"@"**
- comporte au moins un caractère **"."** rencontré après le caractère **"@"**

Prototype : void check_email(void);

Exemple :

```
Terminal
~/W-WEB-024>
# avec comme valeur d'email "pedago-ambitionfeminine@epitech.eu"
~/W-WEB-024> php ex_20.php
courriel valide
~/W-WEB-024>
# avec comme valeur d'email "je troll@les_pangolins"
~/W-WEB-024> php ex_20.php
courriel invalide
~/W-WEB-024>
```



preg_match