W1- Piscine PHP

W-WEB-024

Jour 04

Opérateurs et structures conditionelles

v1.0

EPITECH.
L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE

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.
- Avoir créé le dépôt avec BLIH.
- Tenir à jour régulièrement le dépôt.

- Ne laissez jamais votre session ouverte sans surveillance.

Jour 04

Opérateurs et structures conditionelles

Nom du répertoire:	Piscine_PHP_Jour_04
Droits de ramassage:	ramassage-tek
langage:	php
Taille du groupe:	1

- Votre repertoire ne doit pas contenir de fichiers inutiles (fichiers temporaires, ...)
- Vous ne devez pas oublier votre fichier *auteur*, si vous l'oubliez, la moulinette ne pourra pas vous corriger.
- 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 possible de la semaine.

Etape 1

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_01.php`
Restrictions: Aucune

Écrire une fonction qui affiche **"positive"** suivi d'un retour à la ligne si la variable passée en paramètre est supérieure à 0. Sinon elle n'affiche rien.

Prototype: void print_positive(int $value);
Exemple:

```
print_positive(42);
// affiche "positive"
```

Etape 2

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_02.php`
Restrictions: N'utiliser qu'une seule fois le mot-clé **"if"**.

Écrire une fonction qui affiche **"Site interdit aux mineurs."** si la variable passée en paramètre est inférieure à 18, ou **"Vous pouvez entrer !"** si elle supérieure ou égale à 18. Un retour à la ligne sera effectué après chaque affichage.

Prototype: void print_status(int $age);
Exemple:

```
print_positive(42);
/* affiche
Vous pouvez entrer !
*/
```

Etape 3

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_03.php`
Restrictions: N'utiliser qu'une seule fois le mot-clé **"if"**.

Écrire une fonction qui affichera **"Enfantillage..."** pour une variable passée en paramètre inférieure ou égale à 0, **"the answer."** si la valeur est 42, sinon elle affichera **"je peux aller en boite"**.
Un retour à la ligne sera effectué après chaque affichage.

Prototype: void print_age(int $age);
Exemple:

```
print_age(42);
/* affiche
the answer.
*/
```

Etape 4

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_04.php`
Restrictions: Utiliser le mot-clé **"while"**.

Écrire une fonction qui affiche tous les entiers de 0 à la valeur de la variable passée en paramètre et retourner **"true"**.
Si la valeur est négative, la fonction doit afficher **"Boulet !"** et retourner **"false"**.
Un retour à la ligne sera effectué après chaque affichage.

Prototype: bool print_until(int $max);

Etape 5

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_05.php`
Restrictions:

- Interdiction d'utiliser le mot-clé **"if"**
- Obligation d'utiliser le mot-clé **"do while"**

Écrire une fonction qui affiche tous les entiers de la variable passée en paramètre jusqu'à la valeur 42.
Si la valeur est supérieure à 42, elle devra quand même être affichée une fois.
Un retour à la ligne sera effectué après chaque affichage.
La fonction retourne **"true"** si la variable passée en parametre vaut 42, sinon elle retourne **"false"**.

Prototype: bool print_range(int $min);

Etape 6

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_06.php`
Restrictions: Obligation d'utiliser le mot-clé **"for"**.

Écrire une fonction qui retourne une chaîne de caractères composée d'autant de **"piou"** que la valeur de la variable passée en paramètre, suivi d'un retour à la ligne.
Prototype: string get_angry_bird(int $nbr);
Exemple:

```
get_angry_bird(3);
/* retourne
pioupioupiou
*/
```

Etape 7

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_07.php`
Restrictions: Obligation d'utiliser le mot-clé **"foreach"**.

Écrire une fonction qui affiche toutes les valeurs du tableau passé en paramètre, chacune séparée par un retour à la ligne.
Cette fonction retourne le nombre d'elements du tableau.

Prototype: integer aff_array(array $my_array);

Etape 8

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_08.php`
Restrictions: Aucune

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

Prototype: void print_array_with_key(array $my_array);

Etape 9

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_09.php`
Restrictions: Aucune

Créer une fonction qui affiche (par ordre croissant) tous les multiples compris entre 0 et 200 000 du nombre entier donné en paramètre.
Un retour à la ligne sera effectué après chaque affichage.

Prototype: void multiples(int $nb);

Etape 10

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_10.php`
Restrictions: Obligation d'utiliser le mot-clé **"switch"**

Écrire une fonction qui prend un nombre entier en paramètre. Si la valeur est 3, elle affiche **"Les trois freres"**. Si la valeur est 6, elle affiche **"Sixieme sens"**. Si la valeur est 0, elle affiche **"l'homme invisible"**. Pour 23, elle affiche **"Le nombre 23"**. Et pour 28, elle affiche **"28 jours plus tard"**. Sinon, elle affiche **"Je ne connais pas."**. Un retour à la ligne sera effectué après chaque affichage.

Prototype: void print_film_from_nbr(int $nbr);
Exemple:

```
print_film_from_nbr(23);
/* affiche
Le nombre 23
*/
```

Etape 11

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_11.php`
Restrictions: Aucune

Écrire une fonction qui inclut le fichier dont le nom est passé en paramètre.

Prototype: void require_file(string $filename);

Etape 12

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_12.php`
Restrictions: Aucune

Écrire une fonction qui teste les 2 variables passés en paramètre. Si les variables ont la même valeur, elle affiche **"Same value."**. Si les variables ont le même type, elle affiche **"Same type."**. Si les variables ont le même type et la même valeur, elle affiche **"Same type and value."**. Un retour à la ligne sera effectué après chaque affichage.

Prototype: void is_similar(mixed $a, mixed $b);

Etape 13

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_13.php`
Restrictions:

- Interdiction d'utiliser les mots-clés **"if"** et **"else"**
- Obligation d'utiliser les ternaires

Écrire une fonction qui affiche **"Majeur"** si la valeur de la variable passée en paramètre est supérieure ou égale à 18. Sinon elle affiche **"Mineur"**.

Prototype: void is_major(int $age);

Etape 14

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_14.php`
Restrictions:

- Obligation d'utiliser le mot-clé **"goto"**
- Interdiction d'utiliser **"do while"**, **"while"**, **"for"**
- Interdiction d'utiliser plus d'une fois **"echo"**

Écrire une fonction qui affiche 42 fois le mot **"wac"**, suivi d'un retour à la ligne à chaque fois.

Prototype: void goto_is_evil(void);

Etape 15

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_15.php`
Restrictions: Obligation d'utiliser le mot-clé **"foreach"**

Écrire une fonction qui remplace toutes les valeurs du tableau passé en paramètre par la chaîne de caractères **"rain"**.

Prototypes: void make_it_rain(array &$my_array);

Etape 16

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_16.php`
Restrictions: Obligation d'utiliser le mot-clé **"foreach"**.

Écrire une fonction qui affiche toutes les valeurs du tableau à deux dimensions passé en paramètre. Un retour à la ligne sera effectué après chaque affichage.

Prototypes: void print_double_array(array $my_array);
Exemple:

```
print_double_array(array(array(1, 2.7, 5), array("un", "foo", "bar")));
/* affichera :
1
2.7
5
un
foo
bar
*/
```

Etape 17

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_17.php`
Restrictions: Aucune

Créez une fonction **"get_array_key"** qui devra retourner la clé de l'élément du tableau dont la valeur est passée en paramètre.

Prototype: mixed get_array_key(array $arr, mixed $value);

Etape 18

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_18.php`
Restrictions: Aucune

Créez une fonction **"return_calls"** qui ne prend aucun paramètre et qui retourne le triple du nombre de fois qu'elle est appelée.

Prototypes: int return_calls(void);
Exemple:

```
return_calls();
return_calls();
echo return_calls();
// Le retour du 3eme appel a la fonction devra retourner : 9
```

Etape 19

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_19.php`
Restrictions:

- Interdiction d'utiliser le mot-clé **"continue"**
- Obligation d'utiliser le mot-clé **"break"**

Écrire une fonction qui affiche ce qui est lu sur l'entrée standard. La fonction s'arrête après avoir rencontré et affiché la ligne "EOF".

Prototype: void break_cat(void);

Etape 20

1 points

Nom de rendu: `Piscine_PHP_Jour_04/ex_20.php`
Restrictions:

- Interdiction d'utiliser le mot-clé **"break"**
- Obligation d'utiliser le mot-clé **"continue"**

Écrire une fonction qui affiche ce qui est lu sur l'entrée standard. La fonction s'arrête après avoir rencontré et affiché la ligne "EOF".

Prototype: void continue_cat(void);