PHP #3

SOUS-PROGRAMMES

&

Initiation à la programmation OBJET

# Contexte des exercices

1. Télécharger depuis *Moodle* les fichiers de départ et les décompresser dans un dossier « php » dans votre espace personnel.

Une feuille de style style.css basique s’applique à ce fichier. Vous pouvez appliquer une feuille de style faite par vos soins.

Pour cette séance, vous ferez les exercices demandés dans le fichier **exercices3.php**, avec l’éditeur de code de votre choix. Rappelez-vous qu’**il faut téléverser le fichier à chaque modification** pour le tester.

# Fonctions php

Le langage php propose plus de 4 000 fonctions (1 500 disponibles pour tous les développeurs, généralement un maximum de 200 sont utilisées).

*Ressource Web :* <http://php.net/manual/fr>

Il est important de savoir lire et utiliser la documentation d’une bibliothèque d’un langage de programmation.

1. On a précédemment utilisé le sous-programme **count**. Le rechercher dans la documentation pour identifier sa signature (i.e. l’ensemble des informations nécessaires pour l’utiliser) :

|  |  |
| --- | --- |
| Fonction ou procédure ? | Fonction |
| Type et signification de la valeur de retour : | Type : entier  Valeur de retour : le nombre d’éléments d’un tableau |
| Le type et le rôle du paramètre #1 (et s’il est obligatoire ?) : | Type : tableau  Rôle : tableau à compter |
| Le type et le rôle du paramètre #2 (et s’il est obligatoire ?) : | (Optionnel) |
| … |  |

Dans le fichier d’origine exercices3.php, ce sous-programme est appelé :

echo count($livre);

1. Expliquer :

|  |  |
| --- | --- |
| Comment peut-on dire que le sous-programme est appelé en tant que fonction ou procédure : | Fonction car parenthèses et paramètre(s) dans les parenthèses et car elle retourne une valeur. |
| La signification de la valeur de retour pour cet appel :  Et confirmer si cette valeur de retour est bien du type annoncé : | C’est le nombre d’éléments du tableau passé en paramètre. Elle retourne bien 12. |
| Le paramètre #1 effectif : | $livre |
| Le paramètre #2 effectif : |  |
| … |  |

1. Avez-vous besoin de savoir comment ce sous-programme est codé pour l’utiliser ? Non

Certaines chaînes de caractères provenant de la base de données ne s’affichent pas correctement, à cause des jeux de caractères qui peuvent différer.

1. Rechercher dans la documentation la fonction **utf8**\_**encode** pour identifier sa signature :

|  |  |
| --- | --- |
| Fonction ou procédure ? | Fonction |
| Type et signification de la valeur de retour : | Type : string  Valeur de retour : Paramètre après encodage (string) |
| Le type et le rôle du paramètre #1 (et s’il est obligatoire ?) : | Type : string  Rôle : utiliser une chaine de caractère dans le traitement de la fonction |
| Le type et le rôle du paramètre #2 (et s’il est obligatoire ?) : |  |
| … |  |

1. L’appeler pour afficher correctement le titre, ainsi que le résumé du livre $livre

Même s’il vaut mieux quand c’est possible trier les données en SQL lors de leur consultation en BD, il est possible d’effectuer des tris en php.

1. Rechercher dans la documentation la fonction **sort** pour identifier sa signature :

|  |  |
| --- | --- |
| Fonction ou procédure ? | Fonction |
| Type et signification de la valeur de retour : | Type : booléen  Valeur de retour : true en cas de succès ou false si une erreur survient |
| Le type et le rôle du paramètre #1 (et s’il est obligatoire ?) : | Type : array  Rôle : utiliser un tableau dans le traitement de la fonction |
| Le type et le rôle du paramètre #2 (et s’il est obligatoire ?) : |  |
| … |  |

1. L’appeler pour trier $tabThemes, avant son affichage.

***Remarque****: Même si c’est bien une fonction, sort peut être utilisé - s’appeler - comme une procédure.*

# Fonctions utilisateur

Au-delà des fonctions proposées par le langage php, tout développeur peut écrire ses propres sous-programmes (appelés alors fonctions utilisateur). C’est même recommandé pour écrire des algorithmes performants, réutilisables, maintenables.

1. Dans le fichier de départ, un sous-programme afficheNbPages est déclaré. Observer les commentaires pour documentation pour identifier sa signature :

|  |  |
| --- | --- |
| Fonction ou procédure ? | Fonction |
| Type et signification de la valeur de retour : | Type : string  Valeur de retour : affiche le nombre de pages au format « n p. » |
| Le type et le rôle du paramètre #1 (et s’il est obligatoire ?) : | Type : entier  Rôle : l’utiliser pour afficher un résultat |
| Le type et le rôle du paramètre #2 (et s’il est obligatoire ?) : |  |
| … |  |

1. Le sous-programme est bien déclaré, mais est-il exécuté ? Pourquoi ?

Non car on ne l’a pas appelé.

1. L’appeler pour afficher sous cette forme le nombre de pages (nbpages) du livre $livre
2. Écrire la fonction utilisateur calculant et renvoyant le prix après réduction en fonction du prix et de la réduction passés en paramètres.

***Consigne****: Renseigner les commentaires pour documentation en vous référant au sous-programme précédent.*

1. L’appeler pour le livre $livre
2. Écrire une procédure affichant une ligne composée par plusieurs fois le même caractère (cf. séance #1), ce caractère ainsi que le nombre de caractères étant passés en paramètres.

***Consigne****: Renseigner les commentaires pour documentation.*

1. L’appeler pour faire afficher une ligne de 100 caractères « : ».

# Objet représentant une ligne d’une base de données

De plus en plus de langages web sont utilisés dans le cadre de la **POO (Programmation Orientée Objet)** – PHP l’est également. Parallèlement les bases de données restent majoritairement relationnelles. Il faut alors gérer la correspondance objet <-> relationnel (ORM : Object-Relational Mapping).

En POO, le principe d’**encapsulation** doit être respecté :

Un objet…

* Possède un **état** représenté par les valeurs de ses **attributs (un attribut est une variable caractérisant un objet)**,
* Et propose des **services** utilisables par des **méthodes** (une méthode est une fonction associée à un objet)

Par exemple la variable $livreObjet est un objet dont les attributs sont les valeurs des colonnes d’un livre en base de données.

Conformément à l’encapsulation, on doit passer par une méthode (appelée alors **accesseur**) pour accéder à ses attributs.

1. Observer l’instruction qui affiche le titre du livre $livreObjet, et faire de même pour afficher :
   * Le prix de $livreObjet
   * La réduction
   * La disponibilité
   * Le numéro d’édition
   * La langue

# *Pour aller plus loin…*

1. Appeler de nouveau la fonction afficheNbPages pour l’affichage du nombre de pages du deuxième livre $livreObjet

La fonction **date** affiche, à partir d’une date au format anglo-saxon, cette date dans un format défini.

1. Appeler cette fonction pour afficher la date de parution du deuxième livre ($livreObjet‑>getParution()) dans le format français «  *d/m/Y ».*
2. Appeler la fonction php date\_default\_timezone\_set pour définir le fuseau horaire à utiliser dans vos programmes.
3. Écrire le sous-programme calculant, pour un livre, le prix final en tenant compte de la réduction, puis des frais de port passés en paramètre.
4. Rechercher dans la documentation les fonctions de débogage print\_r et var\_dump pour identifier leur signature.
5. Modifier la méthode getTitre (dans le fichier livre.class.php) pour renvoyant le titre du livre en utf8. Faire de même avec la méthode getResume
6. Modifier la méthode getParution (dans le fichier livre.class.php) pour renvoyant la date de parution du livre dans le format français «  jj/mm/aaaa ».
7. Déplacer la fonction calculant le prix après réduction dans la déclaration des objets Livre dans le fichier livre.class.php (on appelle « **Classe** » la représentation de tous les objets du même « type »). La modifier pour en faire une méthode de cette classe.