

La manipulation des dates

Table des matières

I. Contexte	3
II. Fondamentaux des Dates PHP	3
A. La méthode Date & ses caractères	3
B. Les autres méthodes dates PHP	5
C. Exercice : Quiz	8
III. Calculer et Comparer des dates	8
A. Calculer et Comparer des dates.....	8
B. Exercice : Quiz.....	10
IV. Essentiel	12
V. Auto-évaluation	13
A. Exercice	13
B. Test.....	14
Solutions des exercices	15

I. Contexte

- Durée : 1 h
- Prérequis : les bases en PHP, Programmation Orienté Objet
- Environnement de travail : PHP 8.2 , Programiz PHP online compiler

Contexte

Dans le langage PHP, la méthode `date` permet de retourner une chaîne de caractères représentant une date ou une heure précise. Les dates sont incontournables en développement web, car elles permettent de générer des alertes, de planifier des tâches, d'effectuer des calculs ou des modifications dynamiques. Notamment dans les CMS, tel que WordPress, où développer en PHP pour les blogs, il est possible de planifier la publication d'un article en définissant sa date de publication. On peut voir aussi dans l'e-commerce que les dates sont utilisées notamment pour mettre en promotion un produit pendant un certain temps. Le produit reprend son prix initial une fois la date d'expiration atteinte. La date est aussi utilisée dans les bases de données, entre autres pour enregistrer la date d'inscription d'un utilisateur. La date est une donnée importante pour connaître l'ancienneté d'une personne inscrite, une information utile pour la fidélisation d'un client.

La date est surtout utilisée dans les formulaires, notamment pour les sites de réservations. Le principe est avant tout d'envoyer deux dates au serveur : une date de départ et une date d'arrivée. Le code PHP permet de calculer le nombre de jours entre les deux dates afin de retourner le prix total de la réservation.

Le but de ce cours est de vous apprendre les méthodes les plus utilisées dans la manipulation des dates en PHP, mais aussi comment les insérer dans une base de données.

II. Fondamentaux des Dates PHP

A. La méthode Date & ses caractères

Définition

La méthode Date

En PHP, les dates se renseignent sous deux formats : un format `string` comme chaîne de caractère ou un format `object` appelé `Datetime`. La méthode `date` permet de retourner une date ou heure en une chaîne de caractère. Vous pouvez définir vous-même le format de la date, grâce à des caractères spéciaux énumérés sur le tableau ci-dessous.

Tableau des caractères dates

Pour les paramètres, il faut insérer les lettres suivantes ainsi que les symboles de syntaxes (`/`, `-`) pour définir le format de votre date :

Caractères	Description	Résultat
d	Il correspond au jour avec un zéro initial pour les 9 premiers jours	Nombres entre 01 et 31
j	Il correspond au jour sans zéro initial	Nombres entre 1 et 31
z	Il correspond au jour dans l'année	Nombres entre 1 et 365 (ou 366 si l'année est bissextile)
l	(Lettre L en minuscule) correspond au jour hebdomadaire en anglais	Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday

Caractères	Description	Résultat
m	m (month) correspond au mois au format numérique	Nombres entre 01 et 12
F	Correspond au mois dans un format texte	January, February, March, April, May, June, July, etc.
t	t correspond au nombre de jours dans le mois (exemple pour janvier = 31)	Nombres entre 1 et 31 (Selon le mois)
n	Correspond au format numérique du mois sans 0 initial pour les 9 premiers mois	Nombres entre 1 et 12
L	Retourne si l'année est bissextile	1 (si l'année est bissextile) sinon 0
Y	Correspond à l'année en 4 chiffres	2023 (l'année d'aujourd'hui)
g	Correspond à l'heure au format 12 h	Nombres entre 00 et 12
G	Correspond au format 24 h sans les zéros initiaux	Nombres entre 0 et 24
H	Heures au format 24 h	Nombres entre 00 et 24
i	Pour les minutes avec 0 en initial	Nombres entre 00 et 59
s	s pour les secondes avec 0 en initial	Nombres entre 00 et 59
y	Correspond à l'année en 2 chiffres	23 (L'année d'aujourd'hui)

Exemple Manipulation de la méthode date

```

1 <?php
2 $date3 = date("H:i:s");
3 $date4 = date("Y-m-d H:i:s");
4 $date1 = date("m.d.y");
5 $date2 = date("Ymd");
6 $date5 = date("d l F Y");
7 $date6 = date("L");
8
9 if($date6!=0){
10     $responds="Oui";
11 }else{
12     $responds="Non";
13 }
14
15
16 echo "date 1 :". $date1;
17 echo "date 2 :". $date2;
18 echo "date 3 :". $date3;
19 echo "date 4 :". $date4;
20 echo "date 5 :". $date5;
21 echo "date 6 : l'année est-elle bissextile ? ". $responds;
22 ?>

```

Dans ce morceau de code, la méthode `date` retourne un résultat en fonction du paramètre donné.

La première ligne affichera la date 1 : le mois avec un 0 en initial, le jour avec un 0 en initial, l'année sur 2 chiffres.

La deuxième ligne affichera la date 2 : l'année actuelle sur 4 chiffres, le mois avec un 0 en initial et le jour avec un 0 en initial.

La troisième ligne affichera la date 3 : l'heure, les minutes, les secondes.

La quatrième ligne affichera la date 4 : l'année sur 4 chiffres, le mois avec un 0 en initial, le jour avec un 0 en initial.

La cinquième ligne affichera la date 5 : le jour avec un 0 en initial, le nom du jour en anglais (langue par défaut du serveur), le mois en anglais et l'année sur 4 chiffres.

La sixième ligne affichera la date 6 : il teste si l'année 2023 est bissextile, `$date6` retourne 0, ce qui veut dire que `$respond` = "Non". L'année 2023 compte 365 jours.

B. Les autres méthodes dates PHP

Définition

Le langage PHP permet de faire appel à des méthodes grâce à sa grande bibliothèque. Comme pour les dates, vous pouvez utiliser différentes méthodes pour des actions bien précises. Parmi ces méthodes, nous retrouvons notamment la méthode `time()`.

La méthode `time()` affiche l'horodatage UNIX actuel. Au début des années 1970, le système d'exploitation UNIX a été développé, le prédécesseur de Linux d'aujourd'hui. Dès le départ, il s'est avéré nécessaire de lui donner une représentation du temps, pour laquelle une méthode extrêmement simple a été choisie : un mot double serait utilisé pour mémoriser les soixantièmes de seconde écoulées depuis le 1^{er} janvier 1970.

```
1 time(): int
```

L'horodatage Unix est une unité de temps se calculant en fonction du nombre de temps écoulé en seconde depuis la date du 1^{er} janvier 1970 00 : 00 : 00 UTC tout en respectant la norme POSIX, une norme imposant pour les systèmes UNIX.

Remarque

La méthode `time` n'a pas de paramètres à renseigner.

Exemple

```
1 echo "La date d'aujourd'hui au format Timestamp :".time();
```

La méthode `time` calcule le nombre de secondes qui s'écoule entre la date du 01/01/1970 01:00:00 à la date d'aujourd'hui, ce qui devrait faire 1679663284s.

gmdate

La méthode `gmdate` retourne le format de l'heure complet GMT/GUT. D'après php.net, voici comment nous allons pouvoir rédiger cette méthode :

```
1 gmdate(string $format, ?int $timestamp = null): string
```

GMT « *Greenwich Mean Time* » qui veut dire le temps moyen de Greenwich, il s'agit de l'heure en fonction de la position du Soleil sur le méridien de Greenwich. Greenwich est la ville où se trouve le laboratoire d'observation astronomique.

```
1 echo "La date d'aujourd'hui au format GMT :".gmdate("d/m/Y");
```

La méthode retournera la date d'aujourd'hui au format GMT avec le jour / Mois / Année

getdate()

```
1 getdate(?int $timestamp = null): array
2 source : php.net
```

La méthode `getdate()` affiche la date complète sous forme d'Array List, ce qui permet d'avoir un meilleur visuel de la date sur chaque paramètre. Cependant, `getdate` ne retourne pas des données en string, donc il faut les convertir avec la méthode `print_r()`.

Exemple Exemple de code

```
1 echo print_r(getdate());
```

Il affichera la date d'aujourd'hui sous forme d'Array List, avec les secondes, les minutes, les heures, le jour, le nombre de week-ends, le mois en chiffre, l'année, et le Nième jour dans l'année. Le mois est affiché en anglais et la valeur de la date en Timestamp.

```
1 Array
2 (
3     [seconds] => 4
4     [minutes] => 14
5     [hours] => 14
6     [mday] => 24
7     [wday] => 5
8     [mon] => 3
9     [year] => 2023
10    [yday] => 82
11    [weekday] => Friday
12    [month] => March
13    [0] => 1679667244
14 )
```

La méthode mktime

La méthode **mktime** retourne le timestamp UNIX d'une date. Elle vous permet de créer une date au format Timestamp UNIX à partir des paramètres renseignés.

```
1 mktime(
2     int $hour,
3     ?int $minute = null,
4     ?int $second = null,
5     ?int $month = null,
6     ?int $day = null,
7     ?int $year = null
8 ): int|false
```

Source : [php¹](https://www.php.net/manual/fr/function.mktime.php)

1 <https://www.php.net/manual/fr/function.mktime.php>

Exemple

```
1 date("M-d-Y", mktime(0, 0, 0, 9, 06, 1992));
```

06-09-1992

`mktime` crée une date au format timestamp. Dans cette date, nous avons renseigné l'année 1992, le mois de juin et le jour le 9. La date sera ensuite convertie en chaîne de caractère sur le format Mois numérique - Jour numérique - Année numérique (sur 4 chiffres).

La méthode `gmmktime`

`Gmmktime` retourne le timestamp UNIX d'une date GMT, il suit le même principe de `mktime`.

```
1 gmmktime(  
2     int $hour,  
3     ?int $minute = null,  
4     ?int $second = null,  
5     ?int $month = null,  
6     ?int $day = null,  
7     ?int $year = null  
8 ): int|false
```

Source : [php.net](https://www.php.net)

La méthode `strtotime()`

La méthode `strtotime()` permet de traduire une date de chaîne de caractère à une variable Timestamp.

```
1 strtotime(string $datetime, ?int $baseTimestamp = null): int|false
```

Source : [php.net](https://www.php.net)

Exemple

```
1 echo strtotime("09 June 1992");
```

`Strtotime` va convertir la date en chaîne de caractère en timestamp. Il va pour cela calculer le temps écoulé en seconde entre le 1^{er} janvier 1970 et le 9 juin 1992, ce qui va faire :

708048000

La méthode `checkdate`

La méthode `checkdate` est très utile pour détecter des erreurs de datation, elle vérifie si la date renseignée est dans le calendrier grégorien.

```
1 checkdate(int $month, int $day, int $year): bool
```

Source : [php.net](https://www.php.net)

Exemple

```
1 var_dump(checkdate(2,29 , 2023));  
2 var_dump(checkdate(2, 29, 2024));
```

La méthode `var_dump` permet d'afficher les informations d'une variable.

C. Exercice : Quiz

[solution n°1 p.17]

Question 1

Quel est le résultat time() ?

- ☐ La date d'aujourd'hui en lettre
- ☐ La date d'aujourd'hui au format numérique
- ☐ L'heure, les minutes et les secondes

Question 2

Parmi les méthodes suivantes, lesquelles retournent le résultat 12/11/2023 ?

- ☐ date("d/m/y")
- ☐ date(d/m/Y)
- ☐ date("m/d/Y")
- ☐ date("d/m/Y")

Question 3

Que vaut la méthode suivante : mktime(0, 0, 0, 0, 0, 0) ?

- ☐ 0
- ☐ error
- ☐ 943920000

Question 4

La méthode suivante sera-t-elle supérieure à 0 : mktime(0, 0, 0, 11, 11, 1969) ?

- ☐ Oui
- ☐ Non

Question 5

Quel est le résultat du code ci-dessous ?

```
1 $start = date_create('2023-03-27');
2 $end = date_create('2023-04-02');
3 $nbdays = date_diff($start, $end);
4 echo $nbdays.' jours';
```

- ☐ 6 Jours
- ☐ 0 Jours
- ☐ Fatal error

III. Calculer et Comparer des dates

A. Calculer et Comparer des dates

Dans cette partie, nous allons apprendre 5 cas d'utilisations fréquentes des méthodes date PHP.

Méthode **Ajouter des jours ou des mois**

Pour ajouter des jours ou des mois, utilisez les méthodes `date()` et `strtotime()`. Initialisez votre date au format string. Insérez votre date en ajoutant par concaténation le terme “+1 days” ou “+1 month” dans la méthode `strtotime()` pour ensuite la convertir au format timestamp. Une fois au format timestamp, il ne vous reste plus qu’à convertir le timestamp en chaîne de caractère grâce à la méthode `date()`.

Exemple

```
1 $Date = "2023-03-27";
2 echo date('Y-m-d', strtotime($Date. ' + 1 days'))."|";
3 echo date('Y-m-d', strtotime($Date. ' + 1 months'));
```

2023-03-28|2023-04-27

Méthode **Comment savoir si une date est plus ancienne qu’une autre ?**

Pour comparer deux dates pour retourner la plus ancienne, vous avez plusieurs méthodes à votre disposition : soit comparer directement des date au format string en utilisant les opérateurs ‘<’ ‘>’, soit convertir les dates au format timestamp grâce à la méthode `strtotime()`, soit en utilisant les class `DateTime()` puis utiliser une condition `if` pour savoir si l’une des dates est supérieure à l’autre.

Exemple

```
1 /* Méthode 1 */
2 $Date1 = "2023-03-27";
3 $Date2 = "2023-04-01";
4
5 if($Date1>$Date2){
6
7     echo "la date 1 est plus récente";
8 }else{
9
10     echo "la date 2 est plus récente";
11 }
12
13
14 /* Méthode 2 */
15
16
17 $Date1 = "2023-03-27";
18 $Date2 = "2023-04-01";
19
20 if(strtotime($Date1)>strtotime($Date2)){
21
22     echo "la date 1 est plus récente";
23 }else{
24
25     echo "la date 2 est plus récente";
26 }
27
28 /* Méthode 3 */
29
30 $Date1 = new DateTime("2023-03-27");
31 $Date2 = new DateTime("2023-04-01");
32 $Date1format = $Date1->format("d/m/Y");
33 $Date2format = $Date2->format("d/m/Y");
```

```

34
35 if($Date1 > $Date2){
36     echo "La date 1 : ".$Date1format." est supérieur à la date 2 : ".$Date2format.";
37 }else{
38     echo "La date 2 : ".$Date2format." est supérieur à la date 1 : ".$Date2format.";
39 }
40

```

Méthode Comment calculer l'intervalle de jour entre deux dates ?

Pour calculer le nombre de jours entre deux dates, utilisez les méthodes `date_create` et `date_diff`. `date_create` change la chaîne de caractère de la date au format objet `Datetime`, puis faire la différence entre les deux avec la méthode `date_diff()`. Pour afficher le résultat, il faut le convertir avec la méthode `format`.

Exemple

```

1 $start = date_create('2023-03-27');
2 $end = date_create('2023-04-02');
3 $nbdays = date_diff($start, $end);
4 echo $nbdays->format("%d");

```

Deux dates sont créées du 27 mars 2023 au 04 avril 2023, la méthode `date_diff` calcule la différence de jour entre les deux dates. Si on calcule le nombre de jours qui s'écoule entre ces deux dates, ceci donne 6 jours.

Méthode Comment savoir si un jour est un week-end ?

Pour savoir si une date est un week-end, il suffit d'utiliser la méthode `date` avec le paramètre `N` qui retourne la représentation numérique d'un jour.

Exemple ci-dessous :

```

1 $date = '2022-06-15';
2 $weekendday = date('N', strtotime($date));
3
4 if ($weekendday >= 6) {
5
6     echo 'la date est un weekend';
7
8 } else {
9
10    echo "la date n'est pas un weekend";
11 }
12

```

B. Exercice : Quiz

[solution n°2 p.18]

Question 1

Quel est le résultat du code ci-dessous ?

```

1 $Date1 = "2023-03-27";
2 $Date2 = "2023-03-28";
3 $Date3 = date('Y-m-d', strtotime($Date1. ' + 3 days'));
4 if(strtotime($Date2)>strtotime($Date3)){
5
6     echo "la date 2 est plus récente";
7 }else{
8

```

```
9     echo "la date 3 est plus récente";  
10 }  
11 }
```

- ☐ La date 1 est plus récente
- ☐ La date 2 est plus récente
- ☐ La date 3 est plus récente

Question 2

Quel est le résultat du code ci-dessous ?

```
1 $date = '2023-03-29';  
2  
3  
4 $newdate = date('Y-m-d', strtotime($date. ' + 1 months'));  
5  
6  
7 $weekendday = date('w', strtotime($newdate));  
8  
9  
10 if ($weekendday == 0 || $weekendday == 6) {  
11  
12     echo 'la date est un weekend';  
13  
14  
15 } else {  
16  
17  
18     echo "la date n'est pas un weekend";  
19 }
```

- ☐ La date est un week-end
- ☐ La date n'est pas un week-end
- ☐ Cette méthode n'est pas adaptée à cet exercice

Question 3

Quel est le résultat du code ci-dessous ?

```
1 $start = date_create('2023-05-10');  
2 $end = date_create('2023-05-04');  
3 $nbdays = date_diff($start, $end);  
4  
5  
6 if(strtotime($start)>strtotime($end)){  
7  
8     echo "La date de départ doit être antérieure à la date de fin";  
9  
10 } else {  
11  
12  
13 echo $nbdays->format("%a");  
14 }
```

- ☐ Erreur
- ☐ La date de départ doit être antérieure à la date de fin
- ☐ 6

Question 4

Quelles sont les vraies affirmations par rapport au code ci-dessous ?

```

1 $Date1 = "2023-03-27";
2 $Date2 = "2023-04-27";
3
4
5 $start = date_create($Date1);
6 $end = date_create($Date2);
7 $diffdays = date_diff($start, $end);
8 $nbdays= $diffdays->format("%a");
9
10
11 echo "La liste de toutes les dates:";
12
13
14 for($i=1; $i<=$nbdays; $i++){
15
16     $Date3 = date('Y-m-d', strtotime($Date1. ' + '.$i.' days'));
17     $check=date('N', strtotime($Date3));
18
19     if($check>=6 ){
20
21         echo $Date3."<br/>";
22
23     }
24 }
```

- ☐ Le code calcule le nombre de jours entre deux dates
- ☐ Le code génère une erreur si la date de fin est antérieure à la date de début
- ☐ Le code affiche la liste des dates après le sixième jour entre la date de début et de fin
- ☐ Le code affiche la liste des dates qui tombent un weekend entre la date de début et de fin
- ☐ Le code parcourt toutes les dates entre la date de début et de fin

Question 5

strtotime("0000-0-0") génère une erreur.

- ☐ Vrai
- ☐ Faux

IV. Essentiel

La fonction `date()` renvoie une chaîne formatée représentant la date et l'heure actuelles. Vous pouvez spécifier le format à l'aide de divers caractères de formatage, tels qu'Y pour l'année, m pour le mois, d pour le jour, H pour l'heure, i pour la minute et s pour la seconde. Il y a plusieurs autres méthodes PHP pour manipuler les dates telles que `strtotime()`. Cette fonction analyse une description textuelle datetime en anglais, en un horodatage Unix (le nombre de secondes depuis le 1^{er} janvier 1970 00:00:00 UTC). `mktime()` est utilisée pour créer une date au format

Timestamp. La fonction `time()` renvoie quant à elle l'horodatage Unix actuel (le nombre de secondes depuis le 1er janvier 1970 00:00:00 UTC). Les fonctions `date_create` et `date_diff` permettent de définir l'intervalle entre deux ou plusieurs jours.

Les fonctions `strtotime()` et `date()` peuvent être utilisées pour convertir les dates en chaîne de caractères, puis au format numérique timestamp et vice-versa. Grâce à ces méthodes, on peut ainsi tester l'ancienneté entre deux dates, tester si une date tombe un week-end, calculer le nombre de jours entre deux dates ou encore planifier une action à une date prédéfinie. En combinant les dates en chaîne de caractères et aux formats numériques timestamp, on peut ainsi créer un algorithme qui agit en fonction du temps.

V. Auto-évaluation

A. Exercice

Vous travaillez pour un magasin de grande distribution qui souhaite développer un intranet pour calculer le nombre de jours de congés pris par leurs salariés entre deux dates, en prenant en compte les jours ouvrés du lundi au vendredi et en excluant les jours fériés (sauf les fêtes religieuses comme Pâques).

Question 1

[solution n°3 p.20]

Vous devez développer un code PHP permettant d'effectuer le calcul du nombre de jours de congé en jour ouvré, tout en prenant en compte les jours fériés.

Vous devrez ajouter aussi un système pour savoir si l'employé possède assez de soldes de congés pour les dates données, afin de savoir s'il a droit à ses congés ou non. En cas de refus, le code devra retourner le nombre de jours manquants.

Pour tester votre code, voici les 3 dates à renseigner et leurs résultats attendus, pour un salarié possédant un solde de 5 jours de congés :

- **Date n°1** : 20/03/2023 à 24/03/2023 → 5 jours de congés
- **Date n°2** : 01/04/2023 à 11/04/2023 → 6 jours de congés
- **Date n°3** : 12/07/2023 à 19/07/2023 → 5 jours de congés

Voici la fonction qui permet vérifier si une date est un jour férié en France (à l'exception des fêtes religieuses) :

```
1 function isholiday($timestamp) {  
2  
3  
4 $jour = date("d", $timestamp);  
5 $mois = date("m", $timestamp);  
6 $annee = date("Y", $timestamp);  
7 $EstFerie = 0;  
8 // dates fériées fixes  
9 if($jour == 1 && $mois == 1) $EstFerie = 1; // 1er janvier  
10 if($jour == 1 && $mois == 5) $EstFerie = 1; // 1er mai  
11 if($jour == 8 && $mois == 5) $EstFerie = 1; // 8 mai  
12 if($jour == 14 && $mois == 7) $EstFerie = 1; // 14 juillet  
13 if($jour == 15 && $mois == 8) $EstFerie = 1; // 15 aout  
14 if($jour == 1 && $mois == 11) $EstFerie = 1; // 1 novembre  
15 if($jour == 11 && $mois == 11) $EstFerie = 1; // 11 novembre  
16 if($jour == 25 && $mois == 12) $EstFerie = 1; // 25 décembre  
17 return $EstFerie;  
18  
19  
20 }
```

Question 2

[solution n°4 p.21]

Le magasin, via son site web, aimerait rajouter une fonctionnalité sur son Click & Collect, permettant de baisser automatiquement le prix des articles consommables, qui vont bientôt atteindre la date de péremption, dans le but de limiter le gaspillage alimentaire.

Votre objectif est de créer un algorithme en PHP avec les méthodes dates, permettant diminuer le prix au fur et à mesure que la date de péremption du produit arrive 5 jours avant cette date, soit les 5 derniers jours avant la péremption. Le prix diminuera de 10 % par jour, à compter des 5 derniers jours avant la date de péremption.

La 5ème jour n'étant pas inclus, celui ci n'offre aucune réduction

Par exemple, nous sommes le 17 / 04 / 2023 :

- Une entrecôte initialement au prix de 15€ est consommable, avant le 20/04/2023. Il reste 3 jours avant la date de péremption. Donc le prix est diminué de - 20 % ce qui fera 12 €.
- Un produit alimentaire qui atteint sa date de péremption verra son prix divisé par 2, car il subira une diminution de - 50 %.

Votre objectif est de créer un algorithme qui baisse automatiquement et progressivement de - 10 % par jour le prix d'un produit alimentaire 5 jours avant sa date de péremption.

```

1 <?php
2
3 function reduction($productname,$date,$price){
4
5 $today=date("Y-m-d"); /* Tester le 17/04/2023 */
6 $datestart = date_create($today); //get current server time
7 $dateend = date_create($date);//some future date
8 $diff = date_diff($datestart, $dateend);
9 $daydiff = intval($diff->format("%d"));
10
11 $newprice = $price;
12 if($daydiff<=5){
13 $newprice=$price-($price*(10*$daydiff)/100);
14 }
15 echo "Nom du produit :".$productname."<br>";
16 echo "Prix en promotion :".$newprice."€<br>";
17 }
18 reduction("Laitue",date("Y-m-d", mktime(0, 0, 0, 04, 20, 2023)),5);
19
20 ?>

```

B. Test

Exercice 1 : Quiz

[solution n°5 p.22]

Question 1

Quel est le format d'une date numérique UNIX ?

- ☐ timestamp
- ☐ datetime
- ☐ time

Question 2

Parmi les méthodes suivantes, lesquelles retournent la valeur timestamp d'une date définie ?

- ☐ mktime
- ☐ time
- ☐ gmmktime
- ☐ date

Question 3

Quelle méthode retourne la valeurs suivante : 1 ?

- ☐ mktime(0, 0, 0, 0, 0, 1)
- ☐ mktime(0, 0, 0, 0, 0, 1);
- ☐ mktime(0, 0, 1, 1, 1, 1970);

Question 4

Nous sommes le 27 mars 2023, quel est le résultat de la méthode suivante : date(m-d-y) ?

- ☐ 27-03-23
- ☐ 2023-27-03
- ☐ 03-27-23
- ☐ 03-27-2023

Question 5


Nous sommes le 27 mars 2023, quel est le résultat de la méthode suivante : date(L) ?

- ☐ 1
- ☐ 0
- ☐ 2

Solutions des exercices

Exercice p. 8 Solution n°1**Question 1**


Quel est le résultat time() ?

- ☐ La date d'aujourd'hui en lettre
- ☒ La date d'aujourd'hui au format numérique
- ☐ L'heure, les minutes et les secondes
-  La méthode time retourne une valeur en timestamp.

Question 2

Parmi les méthodes suivantes, lesquelles retournent le résultat 12/11/2023 ?


- ☐ date("d/m/y")
- ☐ date(d/m/Y)
- ☒ date("m/d/Y")
- ☒ date("d/m/Y")

 L'année est sur 4 chiffres donc il faut utiliser le Y majuscule. 12 ou 11 peuvent correspondre soit à un mois ou un jour. Il faut toujours mettre des guillemets pour rédiger la date en chaîne de caractère.

Question 3

Que vaut la méthode suivante : mktime(0, 0, 0, 0, 0, 0) ?


- ☐ 0
- ☐ error
- ☒ 943920000

 Il n'y a aucune erreur à ne rentrer que des 0 en paramètre. De plus, pour créer un mktime égal à 0, il faut créer la date initiale selon la norme POSIX qui est le 1^{er} janvier 1970.

Question 4

La méthode suivante sera-t-elle supérieure à 0 : mktime(0, 0, 0, 11, 11, 1969) ?

- ☐ Oui
- ☒ Non

 Non car d'après la norme POSIX, la date initiale du timestamp est le 1^{er} janvier 1970. Or, dans ce cas, l'année est 1969, le mktime retournera une valeur négative.

Question 5

Quel est le résultat du code ci-dessous ?

```
1 $start = date_create('2023-03-27');  
2 $end = date_create('2023-04-02');  
3 $nbdays = date_diff($start, $end);  
4 echo $nbdays.' jours';
```

- ☐ 6 Jours
- ☐ 0 Jours
- ☒ Fatal error

Q `date_diff` retourne une valeur object qui ne peut pas être affichée via `echo`. Vous pouvez néanmoins convertir le résultat de `date_diff` en string ou en int pour l'afficher.

Exercice p. 10 Solution n°2

Question 1

Quel est le résultat du code ci-dessous ?


```
1 $Date1 = "2023-03-27";
2 $Date2 = "2023-03-28";
3 $Date3 = date('Y-m-d', strtotime($Date1. ' + 3 days'));
4 if(strtotime($Date2)>strtotime($Date3)){
5
6     echo "la date 2 est plus récente";
7 }else{
8
9     echo "la date 3 est plus récente";
10 }
11 }
```

- ☐ La date 1 est plus récente
 - ☐ La date 2 est plus récente
 - ☒ La date 3 est plus récente
- Q** La date 1 est la veille de la date 2, la date 3 est la date 1 + 3 jours donc la date 3 est supérieure à la date 2.

Question 2

Quel est le résultat du code ci-dessous ?

```
1 $date = '2023-03-29';
2
3
4 $newdate = date('Y-m-d', strtotime($date. ' + 1 months'));
5
6
7 $weekendday = date('w', strtotime($newdate));
8
9
10 if ($weekendday == 0 || $weekendday == 6) {
11
12     echo 'la date est un weekend';
13
14
15 } else {
16
17
18     echo "la date n'est pas un weekend";
19 }
```


- ☒ La date est un week-end
- ☐ La date n'est pas un week-end
- ☐ Cette méthode n'est pas adaptée à cet exercice
-  La date est le 29 avril 2023 qui est un samedi, donc la date est bien un week-end.

Question 3

Quel est le résultat du code ci-dessous ?

```

1 $start = date_create('2023-05-10');
2 $end = date_create('2023-05-04');
3 $nbdays = date_diff($start, $end);
4
5
6 if(strtotime($start)>strtotime($end)){
7
8     echo "La date de départ doit être antérieure à la date de fin";
9
10 } else {
11
12
13 echo $nbdays->format("%a");
14 }
```

- ☒ Erreur
- ☐ La date de départ doit être antérieure à la date de fin
- ☐ 6
-  Le résultat du code ci-dessus serait “Erreur” car les variables start et end de la fonction ne sont pas des chaînes de caractères. En effet, la fonction strtotime ne peut prendre comme valeur qu’une chaîne de caractère. Or, ici, nous lui attribuons les valeurs “date_create('2023-05-10')” et “date_create('2023-05-04')”. Pour que cela fonctionne, il aurait fallu que \$start prenne pour valeur '2023-05-10' et que \$end prenne pour valeur '2023-05-04'.

Question 4


Quelles sont les vraies affirmations par rapport au code ci-dessous ?

```

1 $Date1 = "2023-03-27";
2 $Date2 = "2023-04-27";
3
4
5 $start = date_create($Date1);
6 $end = date_create($Date2);
7 $diffdays = date_diff($start, $end);
8 $nbdays= $diffdays->format("%a");
9
10
11 echo "La liste de toutes les dates:";
12
13
14 for($i=1; $i<=$nbdays; $i++){
15
16     $Date3 = date('Y-m-d', strtotime($Date1. ' + '.$i.' days'));
17     $check=date('N', strtotime($Date3));
```


```

18
19     if($check>=6 ){
20
21         echo $Date3."<br/>";
22
23     }
24 }
```

- ☒ Le code calcule le nombre de jours entre deux dates
- ☐ Le code génère une erreur si la date de fin est antérieure à la date de début
- ☐ Le code affiche la liste des dates après le sixième jour entre la date de début et de fin
- ☒ Le code affiche la liste des dates qui tombent un weekend entre la date de début et de fin
- ☒ Le code parcourt toutes les dates entre la date de début et de fin
-  Le code calcule d'abord le nombre de jours entre la date de début et de fin. Grâce aux nombres de jours calculés, le code parcourt les différentes dates via la boucle for en ajoutant chaque fois un jour de plus. Le code vérifie chaque fois si la date tombe un week-end. Le code marche même si la date de fin est antérieure à la date de début.

Question 5

strtotime("0000-0-0") génère une erreur.

- ☐ Vrai
- ☒ Faux
-  Faux, il se traduit par défaut à la date de l'an 1 av. J.-C., le 30 novembre soit -0001-11-30, il vaut -62169984000.

p. 13 Solution n°3

```

1 <html>
2 <head>
3 <title>La manipulation des dates PHP</title>
4 </head>
5 <body>
6 <?php
7     $date1 = '2023-04-01';
8     $date2 = '2023-04-11';
9     $solde = 10; // déclaration et initialisation de la variable $solde
10
11     $datestart = date_create($date1); //get current server time
12     $dateend = date_create($date2); //some future date
13     $diff = date_diff($datestart, $dateend);
14
15     function isholiday($timestamp) {
16         $jour = date("d", $timestamp);
17         $mois = date("m", $timestamp);
18         $annee = date("Y", $timestamp);
19         $EstFerie = 0;
20         // dates fériées fixes
21         if($jour == 1 && $mois == 1) $EstFerie = 1; // 1er janvier
22         if($jour == 1 && $mois == 5) $EstFerie = 1; // 1er mai
23         if($jour == 8 && $mois == 5) $EstFerie = 1; // 8 mai

```

```

24     if($jour == 14 && $mois == 7) $EstFerie = 1; // 14 juillet
25     if($jour == 15 && $mois == 8) $EstFerie = 1; // 15 aout
26     if($jour == 1 && $mois == 11) $EstFerie = 1; // 1 novembre
27     if($jour == 11 && $mois == 11) $EstFerie = 1; // 11 novembre
28     if($jour == 25 && $mois == 12) $EstFerie = 1; // 25 décembre
29     return $EstFerie;
30 }
31 $nbdiff = intval($diff->format("%a"));
32 $conge=0;
33 for($i=0;$i<=$nbdiff;$i++){
34     $newdate = date('Y-m-d', strtotime($date1. ' + '.$i. ' days'));
35     $weekendday = date('w', strtotime($newdate));
36     if ($weekendday == 0 || $weekendday == 6 || isholiday(strtotime($date1. ' + '.$i. '
37     days'))==1) {
38     } else {
39         $conge++;
40     }
41     echo "Nombre de jour de congés : ".$conge."<br>";
42     if($conge>$solde){
43         $diff=$conge-$solde;
44         echo "Congé refusé : solde insuffisant, il manque encore ".$diff." jour(s) de solde";
45     } else {
46         echo "Congé accepté";
47     }
48     ?>
49 </body>
50 </html>

```

p. 14 Solution n°4

```


1 function reduction($productname, $date, $price)
2 {
3     $today = date("Y-m-d"); // Date actuelle
4     $datestart = date_create($today);
5     $dateend = date_create($date);
6     $diff = date_diff($datestart, $dateend);
7     $daydiff = intval($diff->format("%r%a")); // Utiliser %r pour conserver le signe
8     (positif/négatif)
9     if ($daydiff <= 5 && $daydiff >= 0) {
10 // Réduction de 10% par jour pour les 5 derniers jours
11 $reductionPercent = 10 * $daydiff;
12 $newprice = $price - ($price * $reductionPercent / 100);
13 } elseif ($daydiff <=0) {
14 // Le produit est périmé, réduction de 50%
15 $newprice = $price / 2;
16 } else {
17 // Le produit n'est pas encore éligible à la réduction
18 $newprice = $price;
19 }
20
21 echo "Nom du produit : " . $productname . "<br>";
22 echo "Prix en promotion : " . $newprice . "€<br>";
23 }

```

Exercice p. 14 Solution n°5


Question 1

Quel est le format d'une date numérique UNIX ?

- ☒ timestamp
- ☐ datetime
- ☐ time
-  En effet, "*timestamp*" est le format UNIX de la norme POSIX.


Question 2

Parmi les méthodes suivantes, lesquelles retournent la valeur timestamp d'une date définie ?

- ☒ mktime
- ☐ time
- ☒ gmmktime
- ☐ date
-  La méthode "time" ne retourne que la date d'aujourd'hui en timestamp, date retourne la date en chaîne de caractères.


Question 3

Quelle méthode retourne la valeurs suivante : 1 ?

- ☐ mktime(0, 0, 0, 0, 0, 1)
- ☐ mktime(0, 0, 0, 0, 0, 1);
- ☒ mktime(0, 0, 1, 1, 1, 1970);
-  Selon la norme POSIX, le 1^{er} janvier 1970 est la première date. Sa valeur timestamp est de 0 car il s'agit de 0 seconde, si on ajoute 1 seconde, sa valeur timestamp passe à 1, donc mktime(0, 0, 1, 1, 1, 1970) est la bonne réponse.

Question 4

Nous sommes le 27 mars 2023, quel est le résultat de la méthode suivante : date(m-d-y) ?

- ☐ 27-03-23
- ☐ 2023-27-03
- ☒ 03-27-23
- ☐ 03-27-2023
-  m = 03, d = 7, y (minuscule) = 23


Question 5

Nous sommes le 27 mars 2023, quel est le résultat de la méthode suivante : date(L) ?

☐ 1

☒ 0

☐ 2

 L pour préciser si l'année est bissextile, ce qui n'est pas le cas pour l'année 2023.