



Séquence 3

Les boucles-2

LE COURS

Définition



Une structure de contrôle de boucle permet d'exécuter de manière itérative (en boucle) certaines parties du code (bloc de code) tant qu'une condition est vérifiée



Nécessité d'exécuter plusieurs fois à la suite un même code



Une boucle va permettre de n'écrire ce code à exécuter plusieurs fois qu'une seule fois (en fonction d'une condition)



La boucle `while` – "tant que"



La boucle `for` – "pour"



La boucle `do...while` – "faire...tant que"



La boucle `foreach` – "pour chaque"

La boucle

FOR



La boucle **for** va permettre d'exécuter un bloc d'instructions **TANT QU'UNE CONDITION EST VRAI**

```
for (expression1;expression2;expression3) {  
    // instruction(s)  
}
```

Syntaxe

```
for (expression1;expression2;expression3) {  
    // instruction(s)  
}
```



```
for (initialisation;condition;incrémentation) {  
    // instruction(s)  
}
```

For VS While

FOR

```
for (initialisation;condition;incrémentation) {  
    // instruction(s)  
}
```

WHILE

```
// initialisation avant évaluation condition  
while (condition) {  
    // instruction(s)  
    // modification avant évaluation  
}
```



Exemple

```
echo 1. ' ';
echo 2. ' ';
echo 3. ' ';
echo 4. ' ';
echo 5. ' ';
echo 6. ' ';
echo 7. ' ';
echo 8. ' ';
```



Avec une boucle while

```
$nombre = 1;
while ($nombre <= 8) {
    echo $nombre . ' ';
    $nombre += 1;
}
```



Avec une boucle for

```
for($nombre=1;$nombre<=8;$nombre+=1) {
    echo $nombre . ' ';
}
```

Utilisation

```
for (initialisation;condition;incrémentation) {  
    // instruction(s)  
}
```



Peut-être la boucle la plus utilisée !



Principalement utilisée quand on connaît à l'avance le nombre d'itérations

```
for($nombre=1;$nombre<=8;$nombre+=1) {
    echo $nombre . ' ';
}
```

1 itération

Corps de la boucle



On connaît le nombre d'itérations



\$nombre



Variable de boucle

On répète ce code
TANT QUE
la condition est

VRAI

On initialise \$nombre à 1

On évalue la condition

→ Si la condition est **VRAI**



- On affiche \$nombre
- On incrémente \$nombre de 1
- On évalue à nouveau la condition

Incrémentation

L'incrémentation est l'opération qui consiste à **ajouter une valeur** à une **variable**.

\$variable = \$variable + N

\$variable += N

SPECIAL
CASE

Incrémenter une variable de **1**

\$variable = \$variable + 1

\$variable += 1

\$variable++;

++ Opérateur de **post-incrémentation**

Incrémentation

```
for($nombre=1;$nombre<=8;$nombre+=1) {  
    echo $nombre . ' ';  
}
```



+Simple

```
for($nombre=1;$nombre<=8;$nombre++) {  
    echo $nombre . ' ';  
}
```

`$variable++;`**Post-incrémentation**

`$variable = $variable + 1;`
`$variable += 1;`

`$variable--;`**Post-décrémentation**

`$variable = $variable - 1;`
`$variable -= 1;`

LES TRAVAUX PRATIQUES

nombres-pairs-for.php

L'ÉNONCÉ

Ecrire un programme `nombres-pairs-for.php` qui affiche tous les nombres pairs entre 0 et 100.

L'ÉNONCÉ

Modifier le programme `nombres-pairs-for.php` de manière à afficher les nombres pairs entre 0 et un nombre saisi par l'utilisateur.

compte-envers.php

L'ÉNONCÉ | Ecrire un programme **compte-envers** qui compte de **50** à **0** en n'affichant que les nombres de **3** en **3**



```
50 47 44 41 38 35 32 29 26 23 20 17 14 11 8 5 2
```

fizz-buzz.php

L'ÉNONCÉ | Ecrire un programme `fizz-buzz.php` permettant d'afficher les nombre de 1 à 30 en appliquant les règles suivantes :

Si le **nombre** est un **multiple de 3** on affiche **Fizz**

Si le **nombre** est un **multiple de 5** on affiche **Buzz**

Si le **nombre** est un **multiple de 15** on affiche **FizzBuzz**



```
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16 17 Fizz 19 Buzz Fizz 22 23 Fizz Buzz
26 Fizz 28 29 FizzBuzz
```

chiffrement-cesar.php

L'ÉNONCÉ

Écrire un programme **rectangle.php** permettant de dessiner un rectangle avec des "*" dont le largeur et la hauteur sont demandées à l'utilisateur



```
Largeur du rectangle : 10  
Hauteur du rectangle : 6  
* * * * *  
* * * * *  
* * * * *  
* * * * *  
* * * * *  
* * * * *
```

syracuse.php

L'ÉNONCÉ | **Ecrire un programme `syracuse.php` qui génère les 20 premiers termes de la suite de Syracuse d'un nombre donné.**

La règle est la suivante : à partir d'un nombre entier N , on affiche le nombre puis on applique une certaine opération pour obtenir le nombre suivant de la suite.

L'opération est la suivante : si N est pair, on divise par 2; si N est impair, on multiplie par 3 et on ajoute 1.

**Avec $N=15$**

```
Les 20 premiers termes de la suite de Syracuse pour N=15 sont :  
15 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1 4 2
```

**chiffrement-cesar.php**[https://fr.wikipedia.org/wiki/Chiffrement_par_décalage](https://fr.wikipedia.org/wiki/Chiffrement_par_d%C3%A9calage)

L'ÉNONCÉ

Ecrire un programme `chiffrement-cesar.php` qui implémente le chiffrement de César.

Le programme doit demander à l'utilisateur une chaîne de caractères et un décalage (nombre) puis renvoyer la chaîne avec chaque lettre décalée de "décalage" lettres dans l'alphabet.

Par exemple, avec un décalage de 3, "A" devient "D", "B" devient "E", "X" devient "A", etc.

Assurez-vous que le programme gère à la fois les lettres majuscules et minuscules. Les espaces et autres caractères ne doivent pas être modifiés.