

WHILE

php

COURS

+

**TRAVAUX
PRATIQUES**



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)

compte.php

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

1

2

3

4

5

6

7

8

EXAMPLE



On répète plusieurs fois la même instruction



Ce n'est pas bien !

```
echo un_nombre . ' ';
```

Imaginez que l'on veuille compter jusqu'à 100

!



La boucle `while` – "tant que"



La boucle `for` – "pour"



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



La boucle `foreach` – "pour chaque"



La boucle

WHILE



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



```
while (condition) {  
    // instruction(s)  
}
```

TANT QUE la condition est **VRAI**
les instructions sont
exécutées

compte.php

```
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 = $nombre + 1;  
}
```

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

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 **modifie** \$nombre
(ici on **passe** au nombre suivant)
- On **évalue à nouveau** la **condition**



```
// initialisation avant évaluation condition  
while (condition) {
```

```
    // instruction(s)  
    // modification avant évaluation
```

Corps de la boucle



Itération



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

Corps de la boucle



Itération
(ici 8 itérations)



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



Boucle infinie



La variable **\$nombre** est **toujours égale à 1** !



Il faut penser à **modifier** la variable **\$nombre** afin que la **condition** soit vérifiée à nouveau avec une **valeur différente** dans **\$nombre**



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

`$variable = $variable + N`



`$variable += N`

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

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

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



Exécutées **tant**
que la condition
est **VRAI**




Exécutées **lorsque**
la condition est **FAUX**



On **sort** de
la boucle

```
// initialisation
while (condition) {
    // instruction(s)
    if (condition-if) {
        // instructions-if
        break;
    }
    // modification
}
// instruction(s)
```



L'instruction
break permet de
sortir de la
boucle **while**
de **manière**
prématurée

```
// initialisation  
while (condition1 && condition2) {  
    // instruction(s)  
    // modification  
}  
// instruction(s)
```



**Application du
théorème de
De Morgan afin
d'évaluer la
sortie de la
boucle**

```
// initialisation  
while (condition1 || condition2) {  
    // instruction(s)  
    // modification  
}  
// instruction(s)
```



WHILE



nombres-pairs.php

Pair		Impair	
2	4	1	3
6	8	5	7
10		9	

L'ÉNONCÉ

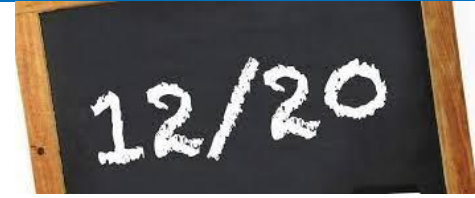
Ecrire un programme `nombres-pairs.php` qui affiche tous les **nombres pairs entre **0** et **100**.**

L'ÉNONCÉ

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



saisie-note.php



L'ÉNONCÉ

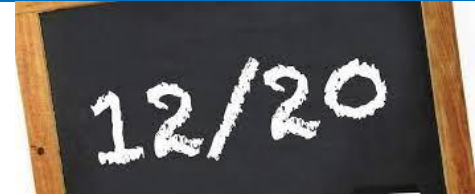
Ecrire un programme **saisie-note.php** qui demande à l'utilisateur de saisir une **note** tant que celle-ci est **strictement supérieure à 20**



```
Saisir une note : 24
La note saisie est incorrecte!
Saisir une note : 21
La note saisie est incorrecte!
Saisir une note : 18
La note saisie est      correcte !
```



saisie-note.php



L'ÉNONCÉ

Modifier le programme **saisie-note.php** afin de demander à l'utilisateur de saisir une **note** tant que celle-ci **n'est pas comprise entre 0 et 20**.



```
Saisir une note : 21
La note saisie est doit être comprise entre 0 et 20 !
Saisir une note : -2
La note saisie est doit être comprise entre 0 et 20 !
Saisir une note : 6
La note saisie est correcte !
```



aleatoire.php



L'ÉNONCÉ

Écrire un programme **aleatoire.php** qui demande à l'utilisateur de saisir un **nombre entre 0 et 1000** et calcule le nombre de coups nécessaire à l'ordinateur pour deviner ce nombre.



Pour faire **générer à l'ordinateur un nombre entre 0 et 1000**, vous utiliserez la fonction PHP suivante :

```
random_int(0,1000)
```



Retourne un nombre entre **0** et **1000**



```
Saisir un nombre entre 0 et 1000 : 215  
Le nombre à deviner a été trouvé en 668 coups
```



aleatoire.php



L'ÉNONCÉ

Modifier le programme **aleatoire.php** afin de vérifier si le **nombre** saisi par l'utilisateur est **compris entre 0 et 1000**



```
Saisir un nombre entre 0 et 1000 : 1200
Le nombre à deviner doit être compris entre 0 et 1000
Saisir un nombre entre 0 et 1000 : 2544
Le nombre à deviner doit être compris entre 0 et 1000
Saisir un nombre entre 0 et 1000 : -45
Le nombre à deviner doit être compris entre 0 et 1000
Saisir un nombre entre 0 et 1000 : 564
Le nombre à deviner a été trouvé en 2028 coups
```

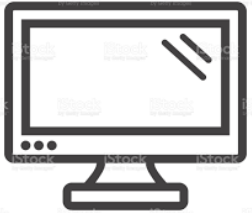


aleatoire.php



L'ÉNONCÉ

Modifier le programme **aleatoire.php** de manière à **mettre fin au calcul du nombre de coups** si le **nombre de coups dépasse le nombre 200**



```
Saisir un nombre entre 0 et 1000 : 5454
Le nombre à deviner doit être compris entre 0 et 1000
Saisir un nombre entre 0 et 1000 : 455
Tu n'as pas deviné en moins de 200 coups !
```



```
Saisir un nombre entre 0 et 1000 : 325
Le nombre à deviner a été trouvé en 152 coups
```