



Séquence 5

Les fonctions

Franck LAMY – BTS SIO1





Les fonctions sont des éléments incontournables dans les langages de programmation

Fonction



Suite d'instructions nommée permettant de réaliser une tâche précise



Fonctions définies par PHP (+ de 1000)



Fonctions définies par l'utilisateur (PHP)





Les fonctions "utilisateur" sont des fonctions créées par les développeurs afin d'identifier et effectuer une tâche précise



Tâche



Suite d'instructions



Les fonctions permettent d'éviter la répétition de code



Plutôt que d'écrire plusieurs fois une même suite d'instructions, on écrit cette suite dans une fonction



Réutilisation du code : lorsqu'une fonction est bien conçue pour effectuer une tâche spécifique, elle peut être réutilisée dans différents endroits du même projet



Les fonctions permettent de structurer son code



Diviser un programme complexe en parties plus petites : le programme est segmenté en fonctions



Facilité de maintenance : si une erreur se produit ou si une mise à jour est nécessaire, vous pouvez vous concentrer uniquement sur la fonction concernée sans avoir à parcourir tout le code



Les fonctions permettent de structurer son code

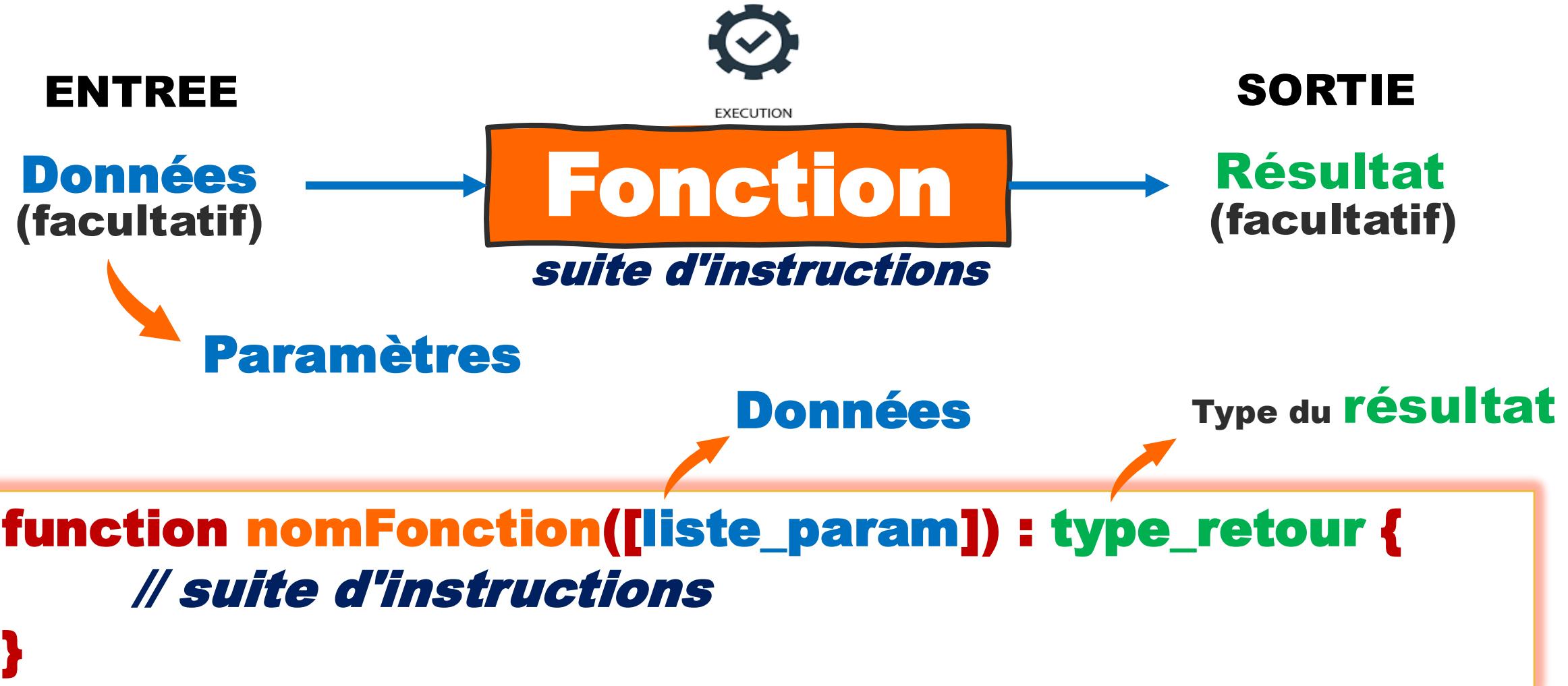


Diviser un programme complexe en parties plus petites : le programme est segmenté en fonctions



Amélioration de la lisibilité : les fonctions aident à rendre le code plus lisible en encapsulant des comportements spécifiques. Cela permet aux autres développeurs (ou à vous-même dans le futur) de comprendre plus facilement ce que fait le programme sans avoir à déchiffrer des blocs de code complexes.

Définir une fonction





=



+



```
function nomFonction([liste_param]*): type_retour
```

{

// suite d'instructions

}



Implémentation de la fonction

*[liste_param] -> FACULTATIF



Une fonction qui retourne un résultat

Fonction

Une fonction qui ne retourne aucun résultat

Procédure



Définir une **fonction** permettant d'afficher
"Bonjour !"

```
function direBonjour() {  
    echo "Bonjour !";  
}
```

```
function direBonjour() : void {  
    echo "Bonjour !";  
}
```

Procédure



Une **fonction** **effectue** une **tâche** **précise**
(action) : on utilise un **verbe** pour la nommer



```
function direBonjour() {  
    echo "Bonjour !";  
}
```



```
direBonjour();
```



L'appel de la fonction va déclencher l'exécution des instructions qui la compose



Une fonction paramétrée est une fonction qui attend en entrée des données (paramètres)

**Liste des paramètres
(type et nom)**

```
function nomFonction(type $p1, type $p2) : type_retour {  
    // suite d'instructions  
}
```

Utilisation des paramètres dans la *suite d'instructions*



```
function nomFonction(type $p1, type $p2) : type_retour {  
    // suite d'instructions  
}
```



Les paramètres sont des variables
UNIQUEMENT CONNUES dans la fonction

variables
LOCALES à
la fonction



"En dehors" de la fonction, les
paramètres n'existent pas !



Définir une **fonction** permettant d'afficher
"Bonjour *prénom*!"

```
function direBonjour(string $prenom) : void {  
    echo "Bonjour $prenom";  
}
```



Le **paramètre \$prenom**
s'utilise comme une **variable**
dans la **fonction**



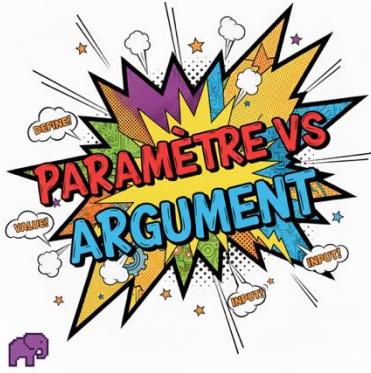
```
function direBonjour(string $prenom) : void {  
    echo "Bonjour $prenom !";  
}
```



```
direBonjour("Jean") ;
```



A l'appel de la fonction, le paramètre \$prenom est créé et initialisé avec la valeur "Jean" → **Argument**

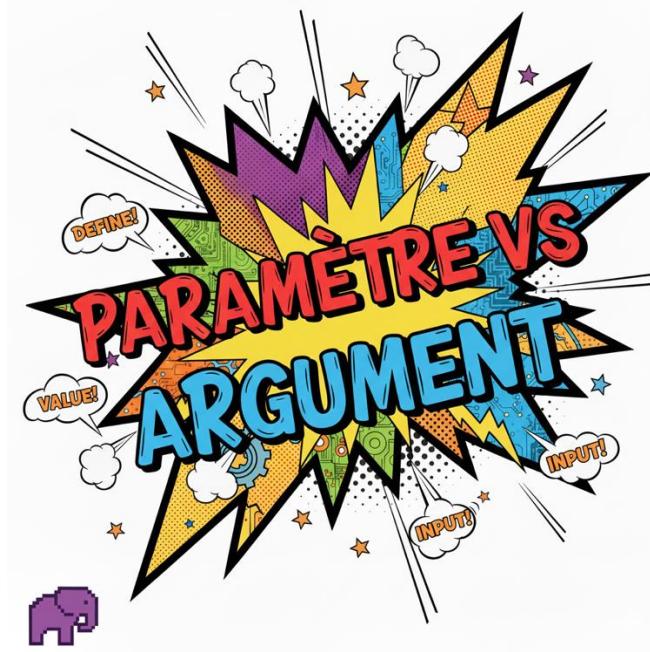


Un **paramètre** est le **nom donné dans la définition de la fonction**

Un **argument** est la **valeur donnée à un paramètre lors de l'appel de la fonction**



Lors de l'appel d'une **fonction** on dit qu'on lui **passe des arguments**



Même type



```
function direBonjour(string $prenom) : void {  
    echo "Bonjour $prenom !";  
}
```



```
$prenom = "Jean";  
direBonjour( $prenom );
```



Le paramètre et l'argument ont le même nom !

Très courant en programmation !



```
function direBonjour(string $prenom) : void {  
    echo "Bonjour $prenom !";  
}
```



```
$prenom = "Jean";  
direBonjour($prenom);
```



COPIE



A l'**appel de la fonction**, le **paramètre \$prenom** est **créé et initialisé avec la valeur de la variable \$prenom (argument)**



fichier.php

```
$prenom = "Jean";
```

// Définition de la fonction

```
function direBonjour(string $prenom) : void {  
    echo "Bonjour $prenom !";  
}
```

// Appel de la fonction

```
direBonjour($prenom);
```

variable GLOBALE
(connue DANS TOUT
le fichier fichier.php)

ATTENTION

Sauf dans les
fonctions

variable LOCALE
(connue UNIQUEMENT dans
la fonction)



Connue = Accessible



fichier.php

```
$prenom = "Jean";
```

// Définition de la fonction

```
function direBonjour() : void {  
    echo "Bonjour $prenom !";  
}
```

// Appel de la fonction

```
direBonjour();
```

variable GLOBALE
(connue DANS TOUT
le fichier fichier.php)



**Sauf dans les
fonctions**

**PHP Warning:
Undefined variable
\$prenom**



**PHP ne sait pas que vous
souhaitez utiliser la
variable GLOBALE
\$prenom**



fichier.php

```
$prenom = "Jean";
```

// Définition de la fonction

```
function direBonjour() : void {  
    global $prenom; -----  
    echo "Bonjour $prenom !";  
}
```

// Appel de la fonction
direBonjour();

variable GLOBALE
(connue DANS TOUT
le fichier fichier.php)

ATTENTION

Sauf dans les
fonctions

Mais...

Indique à PHP que vous
souhaitez utiliser la
variable GLOBALE
\$prenom



fichier.php

```
$prenom = "Jean";
```

```
// Définition de la fonction
function direBonjour() : void {
    global $prenom; -----
    echo "Bonjour $prenom !";
}
```

```
// Appel de la fonction
direBonjour();
```

variable **GLOBALE**
(connue DANS TOUT
le fichier fichier.php)

ATTENTION

Sauf dans les
fonctions



La fonction dépend
d'une variable GLOBALE



Une fonction peut retourner un résultat

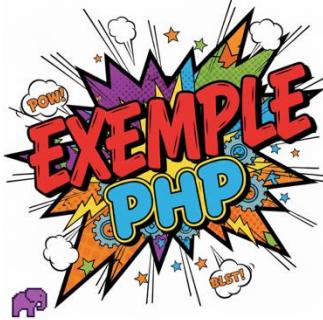


Très fréquent !

```
function nomFonction(type $p1, type $p2) : type_retour {  
    // suite d'instructions  
    return resultat; .....  
}
```



Instruction PHP permettant de retourner un résultat



Définir une fonction permettant de calculer et retourner le résultat de l'addition de 2 nombres

```
function additionner(int $nb1, int $nb2) : int  
{  
    $resultat = $nb1 + $nb2;  
    return $resultat;  
}
```





```
function additionner(int $nb1, int $nb2) : int  
{  
    $resultat = $nb1 + $nb2;  
    return $resultat;  
}
```

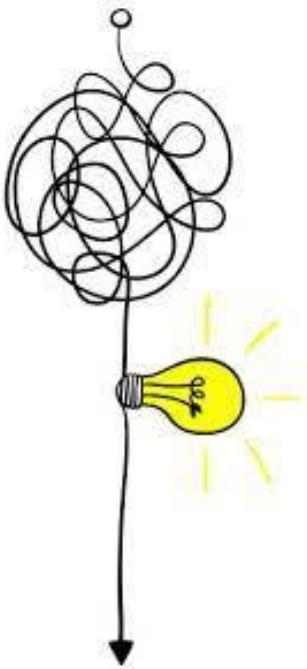


Variable permettant de récupérer le résultat
retourné par la fonction



```
$addition = additionner(12,4);  
echo "Le résultat est $additon";
```





```
function additionner(int $nb1, int $nb2) : int {  
    $resultat = $nb1 + $nb2;  
    return $resultat;  
}
```



```
function additionner(int $nb1, int $nb2) : int  
{  
    return $nb1 + $nb2;  
}
```





```
$addition = additionner(12,4);  
echo "Le résultat est $additon";
```

Utile si on souhaite utiliser le résultat (**variable \$addition**) ultérieurement dans le programme

```
echo "Le résultat est " . additionner(12,4);
```



L'**interpolation** ne fonctionne pas avec l'**appel de fonctions**



Exercice 1

pair.php

L'ÉNONCÉ

Ecrire un programme **pair.php** qui :

- Définit une **fonction** permettant de déterminer si un **nombre** passé en paramètre est **pair** ou **impair**
- Demande à l'utilisateur de saisir son **nombre**
- Affiche si le **nombre saisi** est **pair** ou **impair**



Exercice 2

identite.php



L'ÉNONCÉ

Ecrire un programme identite.php qui :

- **Demande à l'utilisateur de saisir son prénom et son nom**
- **Définit une fonction permettant de retourner sous la forme d'une chaîne de caractères l'identité de l'utilisateur (1^{ère} lettre du prénom en majuscule et nom en majuscules)**

```
Saisir votre prénom : franck
```

```
Saisir votre nom : lamy
```

```
Votre identité est Franck LAMY
```



```
function incrémenter(int $nombre) : void {  
    $nombre += 1;  
}
```



```
$compteur = 1;  
incrémenter($compteur);  
echo $compteur;
```



\$compteur





```
function incrémenter(int $nombre) : void {  
    $nombre += 1;  
}
```

COPIE

```
$compteur = 1;  
incrémenter($compteur);  
echo $compteur;
```



La fonction ne modifie pas \$compteur mais \$nombre qui est une copie

```
function incrémenter(int $nombre) : void {  
    $nombre += 1;  
}
```



```
$compteur = 1;  
incrémenter($compteur);  
echo $compteur;
```

COPIE



Passage des
arguments par
valeur



```
function incrémenter(int $nombre) : int {  
    $nombre += 1;  
    return $nombre;  
}
```


COPIE

```
$compteur = 1;  
$compteur = incrémenter($compteur);  
echo $compteur;
```

APPEL DE LA
FONCTION



```
function incrémenter(int & $nombre) : void {  
    $nombre += 1;  
}
```



LA MEME VARIABLE

```
$compteur = 1;  
incrémenter($compteur);  
echo $compteur;
```



La fonction modifie directement \$compteur



```
function incrémenter(int & $nombre) : void {  
    $nombre += 1;  
}
```



LA MEME VARIABLE

```
$compteur = 1;  
incrémenter($compteur);  
echo $compteur;
```



Passage des
arguments par
référence



Regroupier les fonctions dans des fichiers

fonctions.php

```
function f1() : void { ... }  
function f2(int $nb) : boolean { ... }  
function f3(string $s) : string { ... }
```



fichier.php

```
// Appeler la fonction f2  
$resultat = f2(10);  
echo $resultat;
```



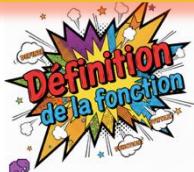
**PHP Fatal error: Uncaught Error:
Call to undefined function f2**



Inclure fonctions.php dans fichier.php

fonctions.php

```
function f1() : void { ... }  
function f2(int $nb) : boolean { ... }  
function f3(string $s) : string { ... }
```



Inclure
(copier/coller)

fichier.php



```
// Appeler la fonction f2  
$resultat = f2(10);  
echo $resultat;
```





Inclure un fichier dans un autre fichier



require ou require_once

include ou include_once



Ctrl+C

Ctrl+V



require ou require_once

fichier.php

```
require 'fonctions.php';
echo "Début du programme";
// Appeler la fonction f2
$resultat = f2(10);
echo $resultat;
```

**Inclus le fichier fonctions.php
s'il existe sinon déclenche une
erreur
(le programme s'arrête)**

fichier.php

```
require_once 'fonctions.php';
echo "Début du programme";
// Appeler la fonction f2
$resultat = f2(10);
echo $resultat;
```

**Idem sauf s'il a déjà
été inclus**



include ou include_once()

fichier.php

```
include 'fonctions.php';
echo "Début du programme";
// Appeler la fonction f2
$resultat = f2(10);
echo $resultat;
```

Inclus le fichier **fonctions.php**
s'il existe sinon déclenche un
warning
(le programme continue)

fichier.php

```
include_once 'fonctions.php';
echo "Début du programme";
// Appeler la fonction f2
$resultat = f2(10);
echo $resultat;
```

**Idem sauf s'il a déjà
été inclus**



Fonction prenant un tableau en paramètre

```
function f1(array $tab) : void {  
    // suite d'instructions  
}
```



```
$tableau = [10,15,18];  
f1($tableau);
```





Fonction prenant un tableau en paramètre

```
function f1(array $stab) : void {  
    // suite d'instructions  
    $stab[0] = 20;  
}
```



```
$tableau = [10,15,18];  
f1($tableau);  
print_r($tableau);
```



[10,15,18]



Toute modification du tableau dans la fonction ne sera pas prise en compte !



Fonction renvoyant un tableau en résultat

```
function f1(...) : array {  
    $tableau = [];  
    // suite d'instructions  
    return $tableau;  
}
```



```
$tab = f1(...);  
foreach($tab as $element) {  
    ...  
}
```

