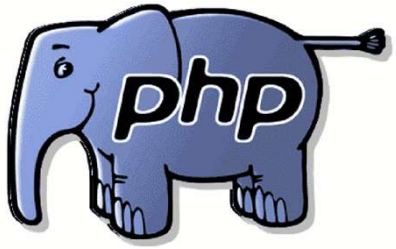
Développeur Web et Web Mobile

Le langage



Partie 1 : Les bases de PhP

PLAN DU COURS

Généralités

Les variables

PLAN DU COURS
suite

Les types de données

Les opérateurs

PLAN DU COURS
suite

Les structures de traitement

Les fonctions

PLAN DU COURS
suite

GENERALITES
présentation

- Le « PHP: Hypertext Preprocessor », est un langage de script libre.
- Utilisé principalement pour créer des pages web dynamiques via un serveur http.
- Peut être utilisé en ligne de commande.
- C'est un langage interprété même si des solutions existent pour gagner en rapidité.
- Il a aujourd'hui les fonctionnalités complètes d'un langage orienté objet.

GENERALITES
caractéristiques

- Simplicité d'écriture de scripts
- Possibilité d'inclure le script PHP au sein d'une page HTML
- Simplicité d'interfaçage avec des bases de données (de nombreux SGBD sont supportés, mais le plus utilisé avec ce langage est MySQL, un SGBD gratuit sur les plates-formes Linux et Windows).
- Intégration au sein de nombreux serveurs web (Apache, Microsoft IIS, ...)
- Gratuité et la disponibilité du code source

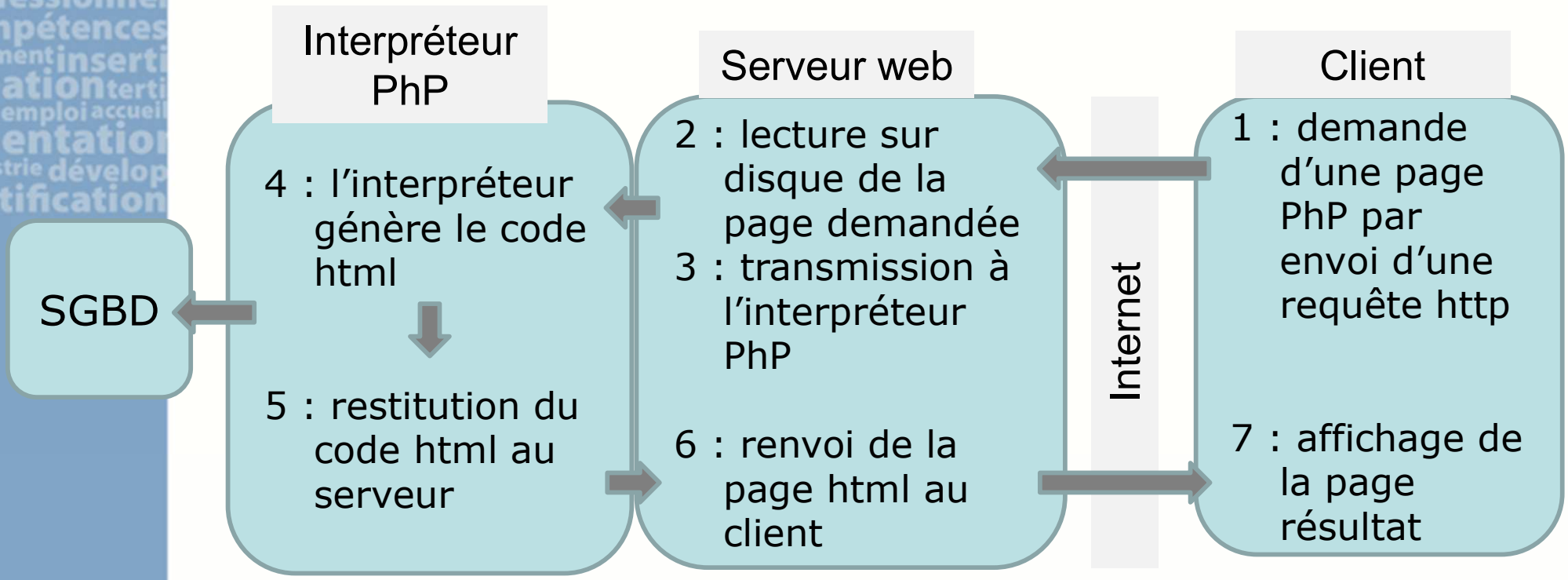
GENERALITES
historique

- Bibliothèque écrite en PERL en 1994 par Rasmus Lerdorf pour son site web :

  Il voulait conserver une trace des personnes qui venaient consulter son CV en ligne.
- Réécriture en C en 1995 et publié sous le nom de "Personal Home Page Tools".
- En 1997 deux étudiants Andi Gutmans et Zeev Suraski redéveloppent le cœur et la version 3 voit le jour sous le nom de Php: Hypertext Processor
- La version 4 suit avec un nouveau moteur appelé « Zend Engine » (**Ze**ev A**nd**i)
- En 2002 c'est la version 5 avec Zend Engine 2 et un véritable modèle objet
- La dernière version est la 5.4.3 sortie en mai 2012
- Une version 6 est en cours

GENERALITES
principe

- Un script PHP est un simple fichier texte contenant des instructions incluses (encapsulées) dans un code HTML à l'aide de balises spéciales.



GENERALITES
passage du html au PhP

- Utilisation de balises spécifiques

  <?php …… **?>**

- Autres notations :
  1. <? ……. ?> (notation racourcie)
  2. <% ….. %> (notation ASP)

     Elles sont déconseillées car peuvent être désactivées dans la configuration du serveur

     Interdit depuis PhP6
  3. <script language="php"> …… </script>

     Peu courante

- En ces 2 balises, autant de lignes php

GENERALITES
syntaxe générale

```
<?php
        echo "Coucou"; // commentaire 1 ligne

        /* commentaire sur plusieurs lignes,
        voilà la deuxième ligne*/

        echo "Ceci est encore un test";
?>
```

Séparation des instructions

Commentaires
//
/* .... */

implique

PHP différencie
MAJUSCULES et minuscules

LES VARIABLES
introduction

- Mémorise une information qui sera manipulée par le programme.
- PhP : langage faiblement typé

  ⟹ pas de déclaration de type
- De nombreuses fonctions existent pour :

  connaitre son type, savoir si elle est vide, …
- Possibilité de modifier le type

  ⟹ « transtypage »

Exemple :

```
$varEntier = 10;
$varChaine = (string)$varEntier;
```

- Une variable peut être : locale ou globale

  ⟹ voir le chapitre sur les fonctions

LES VARIABLES
syntaxe

Commence obligatoirement par « $ »

```
$maChaine = "Coucou";
$monCompteur = 1;
```

IDENTIFICATEUR

- Préférer la notation « **Camel Case** »
- Tout caractère « **alphanumérique** »
- Pas de chiffre en premier caractère
- Pas d'espace
- Doit être « **auto informant** »

$maChaine

≠

$MACHAINE

LES VARIABLES
dynamiques

- Ce sont des variables dont le nom dépend d'autres variables.
- Ce nom est construit dynamiquement pendant exécution du programme.
- Utilisation du double « $ » pour la syntaxe.

Exemple :

```
$mess = "variable dynamique";
$varDyn = "mess" ;
echo $$varDyn; //affiche "variable
                    dynamique"
```

LES VARIABLES
dynamiques (suite)

- Le recours aux variables dynamiques n'est généralement pas nécessaire (remplacées par les tableaux).
- Il est parfois utile d'avoir recours aux accolades.

Exemple :

```
$deuxieme = "contenu de $deuxieme";
$troisieme = "contenu de $troisieme";
$varDyn = "deux";
echo ${$varDyn}ieme;
                    //affiche "contenu de $deuxieme"
$varDyn = "trois";
echo ${$varDyn}ieme;
                    //affiche "contenu de $troisieme"
```

LES TYPES
Introduction

types scalaires

- booléen *(boolean)*
- nombre entier *(integer)*
- nombre à virgule flottante *(float)*
- chaîne de caractères *(string)*

8 types de données

types composés

- tableau *(array)*
- objet *(object)*

types spéciaux

- ressource *(ressource)*
- valeur nulle *(NULL)*

- type le plus simple
- 2 valeurs : **TRUE** ou **FALSE**

Exemple :

```
$bool = TRUE;     // donne la valeur TRUE à $bool
```

- Lors de la conversion de type :

Évalués à FALSE	Évalués à TRUE
Entier 0 ou réel 0.0	Entier ou réel différents de 0
La chaine vide ""	Une chaîne non vide
La valeur NULL	

LES TYPES
Les entiers

- Type
  « **int** »
  ou
  « **integer** »

de **-2 147 483 648**
à **2 147 483 647**

- exprimés dans différentes bases :

entier

$varEnt = 6837;

hexadécimale

$varHex = 0x1AB5;

octale

$varOct = 015265;

($varEnt, $varHex et $varOct exprime la même valeur)

LES TYPES
Les réels

- Type « **double** » ou « real » ou « float »

➢ **dépendent de la plateforme**
➢ **généralement :**
de $-1.8\ 10^{-308}$ à $1.8\ 10^{308}$

- Peuvent s'exprimés :

normale

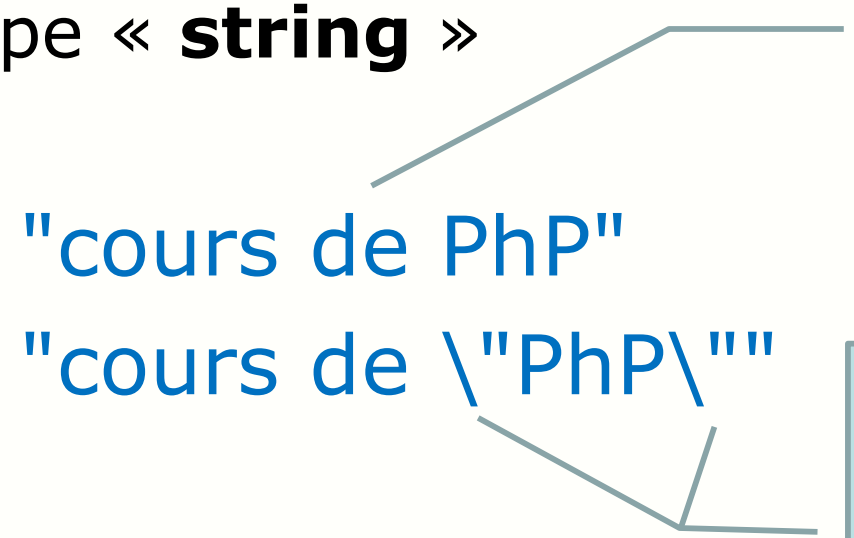
$vReal = 3.14159;

scientifique

$vReal = 2.5e12;

Remarque : un calcul peut donner un résultat indéfinis qui sera repéré par la constante « **NaN** » (Not a Number)

- Type « **string** »



Série de caractères codés sur 1 octet

Echappement des caractères spéciaux en les préfixant par « \ »

- Chaîne littérale : plusieurs syntaxes :
  - Entourée de guillemets simple « ’ »
  - Entourée de guillemets double « " »
  - Syntaxe « heredoc »
  - Syntaxe « nowdoc »

LES TYPES
Les chaînes entre guillemets simples

Echappe uniquement « \' » et « \\ »

```
echo 'L\'affichage est c:\\*.*';
```

L'affichage est c:*.*

```
$prenom = 'Paul';
echo 'Dupond $prenom';
```

Dupond $prenom

Les variables se trouvant à l'intérieur ne sont pas traitées (remplacées par leurs valeurs)

Les chaînes entre guillemets doubles

- Echappe beaucoup de caractères spéciaux

\n	Fin de ligne
\r	Retour à la ligne
\t	Tabulation horizontale
****	Antislash
\$	Signe dollar
\"	Guillemet double

Les variables se trouvant à l'intérieur sont traitées

```
$prenom = "Paul";
echo "Dupond $prenom";
```

Dupond Paul

LES TYPES
Les chaînes syntaxe heredoc

```
$prenom = <<<EOT
    Paul
    EOT;
```

Débute par « **<<<** » suivi d'un **identifiant** puis une **nouvelle ligne**

Se termine par l'identifiant

```
echo "Dupond $prenom";
```

Dupond Paul

- Se comporte comme une chaine entourée de guillemets doubles

- La syntaxe « nowdoc » est identique mais se comporte comme pour les simples guillemets. L'identificateur est à mettre entre guillemets simples.

- Liste d'éléments pas obligatoirement du même type.

Création
- fonction « **array()** »
- PhP 5.4 : directement « **[...]** ».

tableau indexé	**tableau associatif**
suite de « **valeurs** » séparées par des virgules	suite de couples « **clé** » => « **valeur** » séparés par des virgules

- Il est possible de mélanger tableau indexé et associatif.

LES TYPES
Les tableaux (suite)

- En fait PhP ne distingue pas les tableaux indicés et associatifs.
- Si la clé est omise, elle est considérée comme numérique et s'incrémente de 1 à partir de la dernière valeur.
- Les clés peuvent être soumises à un transtypage vers le type entier.

```
"1"     ->   1
"1.5"   ->   1
1.5     ->   1
true    ->   1
```

LES TYPES
Les tableaux: exemples

```
1 : $jourOuvre = array( "Lundi",
                        "Mardi",
                        "Mercredi",
                        "Jeudi",
                        "Vendredi");
        echo $jourOuvre[2];
```

Mercredi

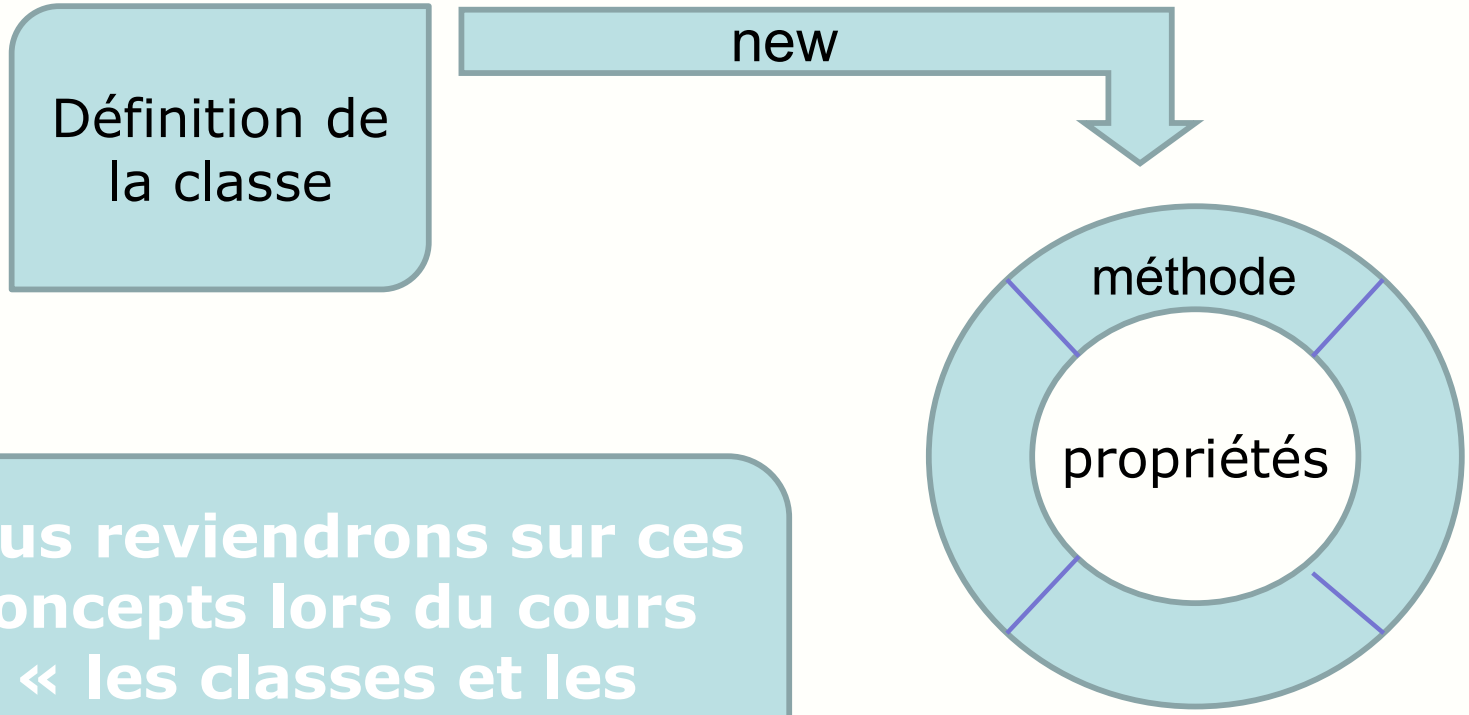
```
2 : $dateNaiss = array(17, "juin", 1998);
    $dateNaiss = [17, "juin", 1998];
        echo $dateNaiss[1];
```

juin

```
3 : $dateNaiss = array( "jour" => 17,
                        "mois" =>"juin",
                        "annee" => 1998);
        echo $dateNaiss["mois"];
```

juin

- Un objet possède des valeurs par ses propriétés et des fonctions par ses méthodes.



Nous reviendrons sur ces concepts lors du cours « les classes et les objets »

LES OPERATEURS
arithmétiques

- Opérateurs élémentaires

-$a	Négation	Opposé de $a
$a + $b	Addition	Somme de $a et $b
$a - $b	Soustraction	Différence de $a et $b
$a * $b	Multiplication	Produit de $a et $b
$a / $b	Division	Quotient de $a et $b
$a % $b	modulo	Reste de la division de $a par $b

- Opérateurs d'in(dé)crémentation

++$a	Pré-incrémentation	$a = $a+1 puis retourne $a
$a++	Post-incrémentation	retourne $a puis $a = $a+1
--$a	Pré-décrémentation	$a = $a-1 puis retourne $a
$a--	Post-décrémentation	retourne $a puis $a = $a-1

LES OPERATEURS
chaînes de caractères

- Opérateur de concaténation « . »

```
$debutChaine = "les opérateurs";
$finChaine = " de chaines";
$chaine = $debutChaine . $finChaine;
echo $chaine;
```

les opérateurs de chaines

- Préférer la concaténation avec les chaines entre guillemets simples plutôt que des variables à l'intérieur des chaines entre guillemets doubles.

```
$finChaine = ' de chaines';
$chaine = "les opérateurs $finChaine";
$chaine = 'les opérateurs' . $finChaine;
```

LES OPERATEURS
d'assignation

Assignation simple
Utilisation du caractère « = »

```
$var = 56;  // assigne 56 à la variable $var
```

Assignation composée
Opérateur suivi du signe d'assignation
Fonctionne avec tout les opérateurs +,-,/,*,%,.

```
$var += 4;  // $var prend la valeur 60
$chaine = "les opérateurs";
$chaine .= " de chaines";
echo $chaine;
```

LES OPERATEURS
de comparaison

- Comparaison entre deux variables

$a == $b	égal	TRUE après transtypage
$a === $b	identique	TRUE si $a même type que $b
$a != $b	différent	TRUE après transtypage
$a <> $b	différent	TRUE après transtypage
$a !== $b	différent	TRUE si $a différent de $b ou si $a pas même type que $b
$a < $b	plus petit	TRUE si $a strictement plus petit que $b
$a <= $b	plus petit ou égal	TRUE si $a plus petit ou égal à $b
$a > $b	plus grand	TRUE si $a strictement plus grand que $b
$a >= $b	plus grand ou égal	TRUE si $a plus grandou égal à $b

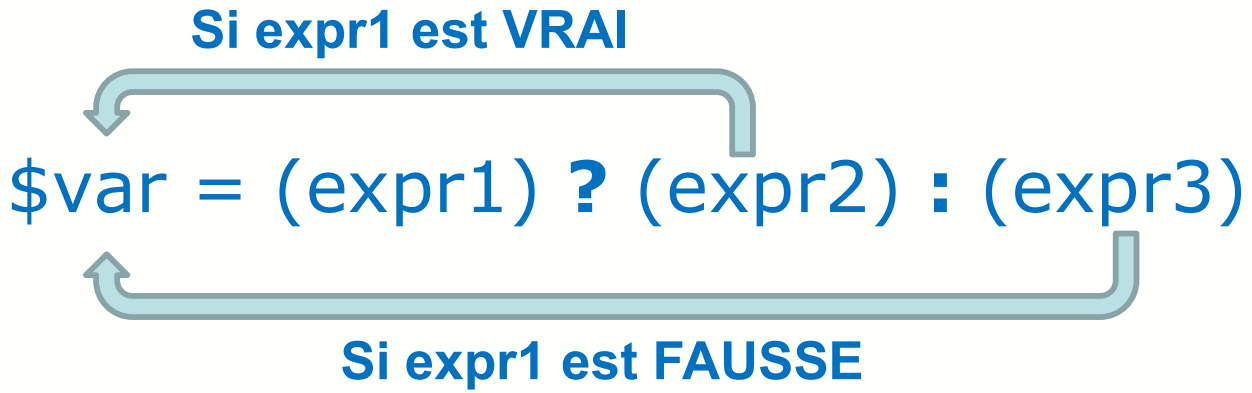
LES OPERATEURS
logiques

- Opérations entre deux variables

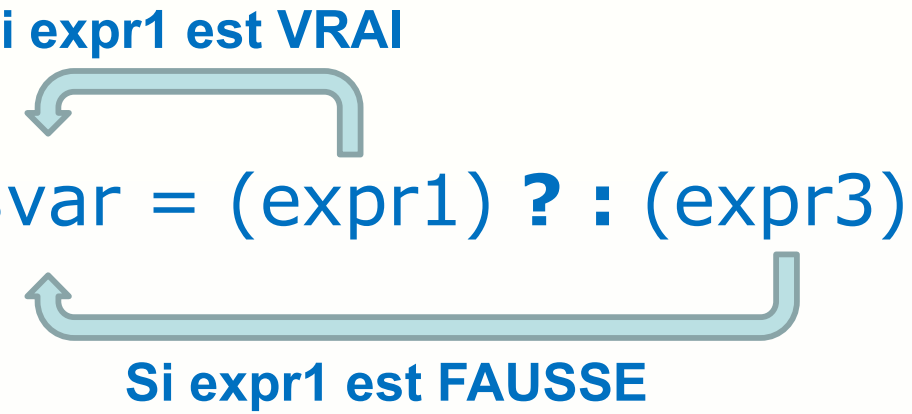
$a and $b	ET	TRUE si $a ET $b valent TRUE
$a or $b	OU	TRUE si $a OU $b valent TRUE
$a xor $b	OU exclusif	TRUE si $a OU $b valent TRUE mais pas les deux
! $a	NON	TRUE si $a n'est pas TRUE
$a && $b	ET	TRUE si $a ET $b valent TRUE
$a || $b	OU	TRUE si $a OU $b valent TRUE

LES OPERATEURS
de test (ternaire)

- Retourne la **valeur** d'une expression en fonction d'une condition.

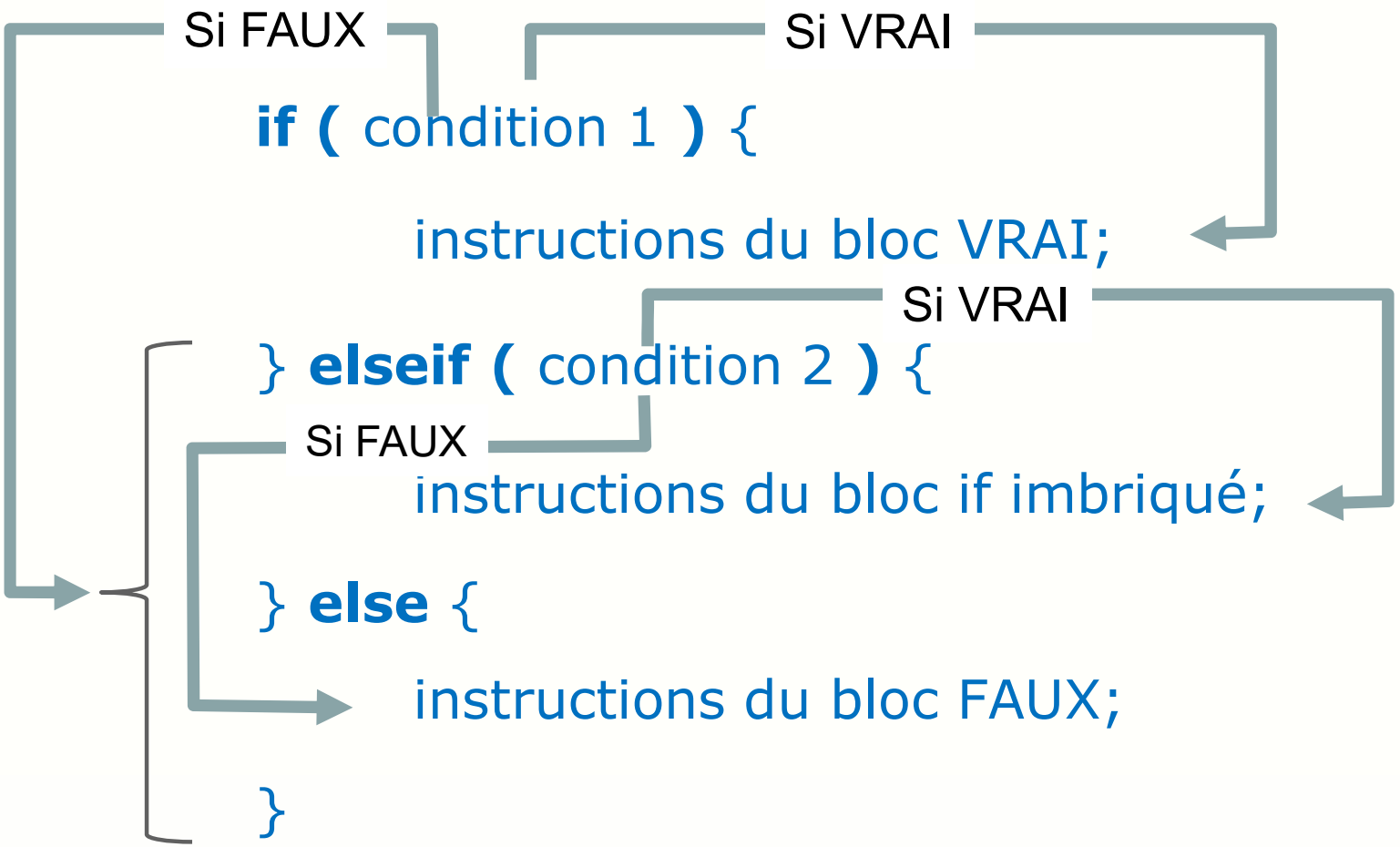


- Expr2 peut être omise :



LES STRUCTURES DE TRAITEMENT
Si ... Alors ... Sinon ...

- Mots clés : **if**, **else**, **elseif**



- Les conditions peuvent être complexes

LES STRUCTURES DE TRAITEMENT
Suivant … Décider … Entre …

- Mot clé : **switch**

Variable numérique entière ou chaine de caractères

```
switch ( variable ) {
        case valeur1 :
                instructions cas valeur1;
                break;
        case valeur2 :
                instructions cas valeur2;
                break;
        ……
        case default :
                instructions cas valeur1;
}
```

LES STRUCTURES DE TRAITEMENT
Tant Que

- Mot clé : **while**
- La syntaxe générale :

```
while ( condition) {
        instructions condition VRAI;
}
```

- Il existe un **do** … **while** :

```
do {
        instructions condition VRAI;
} while ( condition)
```

- Mots clés : **for**

exécutée 1 et 1 seule fois avant toute itération sert à initialiser la variable de boucle

condition évaluée à chaque itération si VRAI réitération si FAUX sortie

```
for( exprInit; condition; exprFin ) {
        instructions;
}
```

exécutée à la fin de chaque itération sert à incrémenter la variable de boucle

LES STRUCTURES DE TRAITEMENT
Pour … Chaque

- Mots clés : **foreach**

tableau (ou objet) sur lequel l'itération se fait

optionnel :
variable recevant le nom de la clé de l'élément du tableau
ou celui de la propriété de l'objet

```
foreach( $tableau as $cle => $valeur ) {
        instructions;
}
```

instructions utilisant la variable $valeur

variable prenant la valeur de chaque élément du tableau ou celui de la propriété de l'objet

- Mots clés : **function**, **return**

```
function nomDeLaFonction( paramètres formels )
{
    instructions;
    return valeur;
}
```

Variable ou constante représentant la valeur de retour de la fonction

liste de variables ou constantes séparées par des virgules

- déclaration possible de variables.

LES FONCTIONS
La portée des variables

```
<?php
$age = 0;
function calculAge($anneeNaiss)
{
   $annee = getDate()["year"];
   return $annee - $anneeNaiss;
}
$monAnnee = 1978;
$age = calculAge($monAnnee);
echo 'mon age est : ' . $age;
?>
```

variables globales

variables locales

LES FONCTIONS
La portée des variables : remarques

- Une fonction accède à une variable globale en la déclarant par le mot clé « **global** »

```
$age = 0;
function calculAge($anneeNaiss)
{
    global $age;
    ……
}
```

- Une variable locale à la durée de vie de la fonction (création à l'appel et destruction).
- Une variable locale à une fonction peut conserver sa valeur entre 2 appels si elle est déclarée par le mot clé « **static** ».

LES FONCTIONS
Le passage de paramètres

- PhP supporte le passage d'arguments
  - par « **valeur** » (comportement par défaut).
  - par « **référence** »

    Utiliser le signe « **&** » devant l'argument dans la déclaration de la fonction
  - les **valeurs d'arguments par défaut**

    Utiliser le signe « **=** » avec la valeur derrière l'argument dans la déclaration de la fonction

    (marche pour arg. par valeur et référence)

    Si tous les arguments non pas de valeur par défaut, mettre ceux-ci en fin de liste.
  - les **listes variables** d'arguments

    Utiliser les fonctions « func_num_args(), func_get_arg(), func_get_args() »

ECRITURE SUR LA CONSOLE
envoi direct

- La fonction « **echo** ».
- Ce n'est pas réellement une fonction.
  (parenthèses non obligatoires)
- Admet une liste d'arguments variables de type « string ».

Exemple :

```
echo 'mon age est : ', $age;
```

Remarques :

1) Passage de 2 arguments
2) si $age est de type numérique, il sera converti en type « string »
3) Il est aussi possible d'utiliser l'opérateur de concaténation

ECRITURE SUR LA CONSOLE
envoi formaté

- La fonction « **printf** »
- Permet une sortie formatée de chaque argument suivant une chaine de formatage
- Admet un nombre variable d'arguments
- La syntaxe est :

printf(chaineFormat, arg1, arg2, …)

chaineFormat est une chaine de caractères dans laquelle sont présents des spécificateur de type qui seront remplacés par les arguments

Exemple :

```
printf("mon age est de %d ans", $age);
```

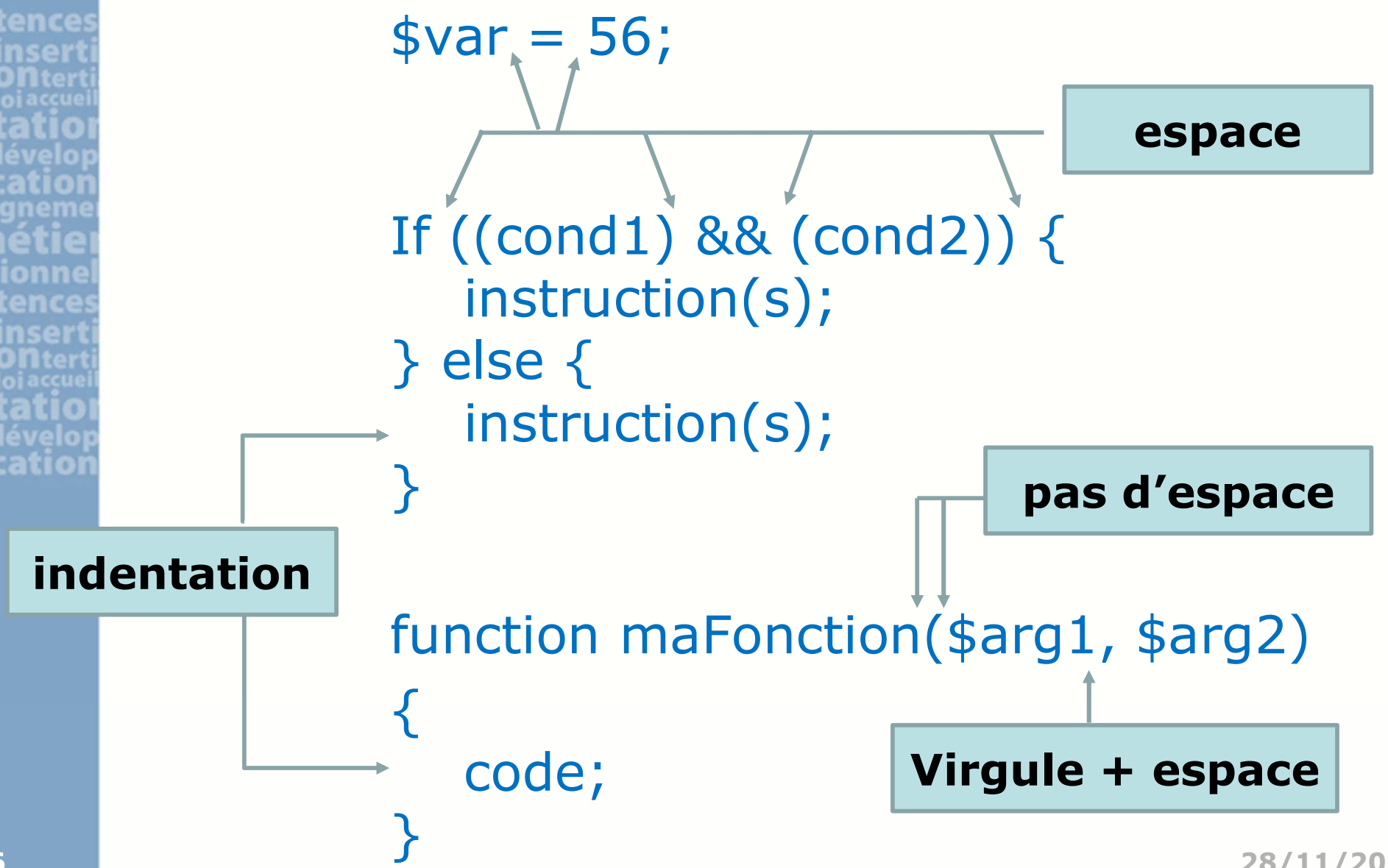
ECRITURE SUR LA CONSOLE
envoi formaté (suite)

- Les principaux spécificateurs de types :

%s	Argument traité comme une chaîne
%c	Argument traité comme un caractère
%d	Argument traité comme un entier
%f	Argument traité comme un réel

- Possibilité de spécifier
  - Le signe du nombre : %+d
  - Le nombre de caractères : %5d
  - Un remplisseur (par défaut espace) : %05d (ici 0, pour autre caractère le mettre entre '')
  - Un nombre de chiffres après virgule : %6.2f

LES REGLES DE CODAGE
présentation du code



LES REGLES DE CODAGE
les identifiants

Majuscule et « _ »

```
const MA_CONSTANTE = 45;
$_maVarGlobale = 56;
```

Minuscule et Camel

```
function maFonction($arg1, $arg2)
{
    $maVarLocale = 0;
}
```

Majuscule et Camel

```
class MaClasse
{
    code;
}
```

LES REGLES DE CODAGE
les constantes

- PhP permet de définir des constantes par la fonction « **define()** ».

Exemple :

```
define("COURS", "PhP");
define("VERSION", 5.3);
```

- L'avantage des constantes est qu'elles sont visibles dans tout le code.
- Nous verrons plus tard, que PhP défini tout un tas d'autres constantes prédéfinies.

(PHP_VERSION, __FILE__, TRUE, …)

Utilisation :

```
echo "la version de php est ".VERSION;
```

LES REGLES DE CODAGE
les constantes (suite)

- Une constante n'admet que des valeurs de type scalaire (voir chapitre "Les types").
- Depuis la **version 5.3** de PhP, il est possible d'utiliser le mot clé « **const** »

Exemple :

```
const COURS = "PhP";
const VERSION = 5.3;
```

  - Attention : il faut déclarer les constantes au plus haut niveau.
  - Ne pas utiliser « const » dans les fonctions, les boucles ou les instructions si … alors

Fin de la partie 1