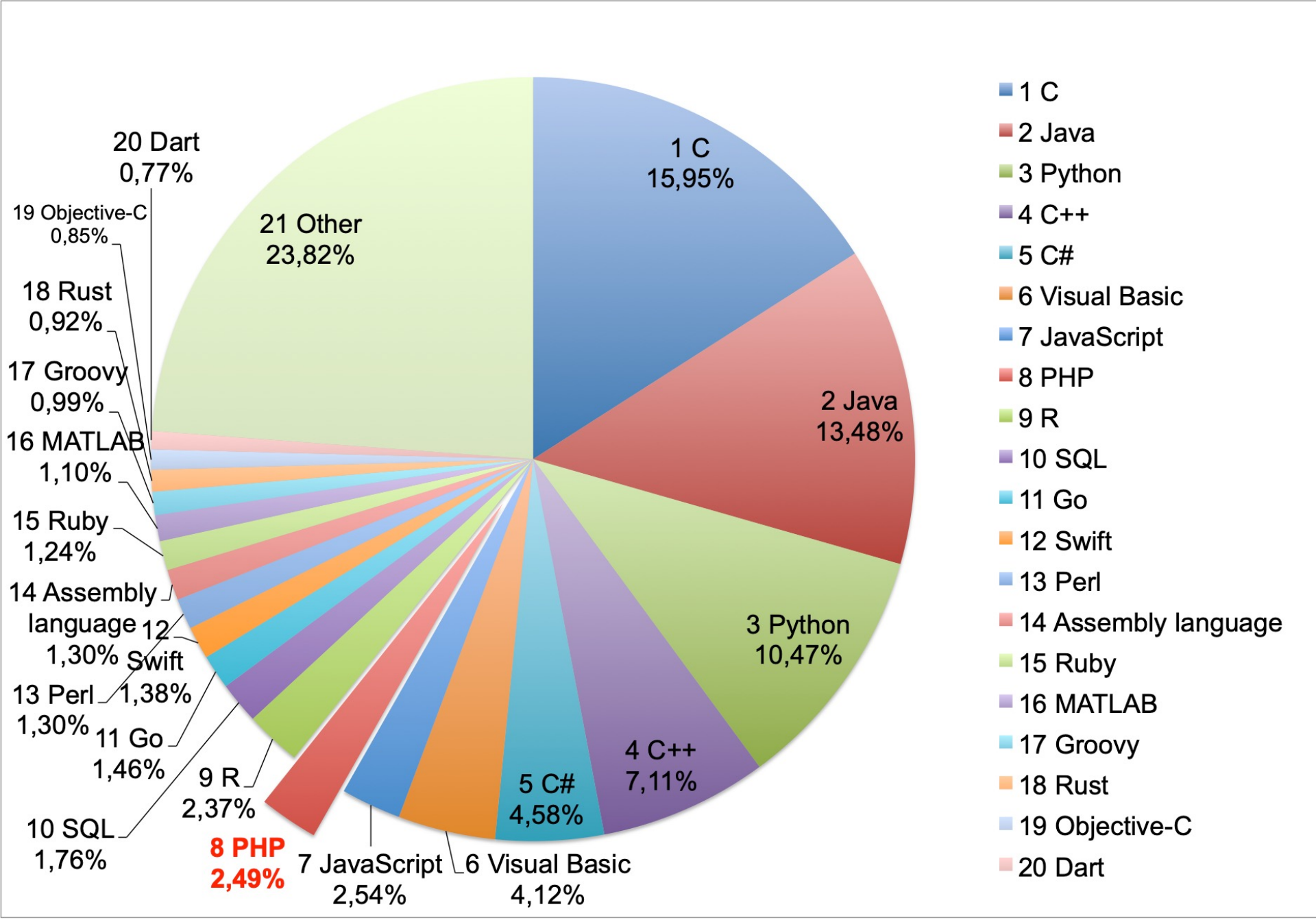
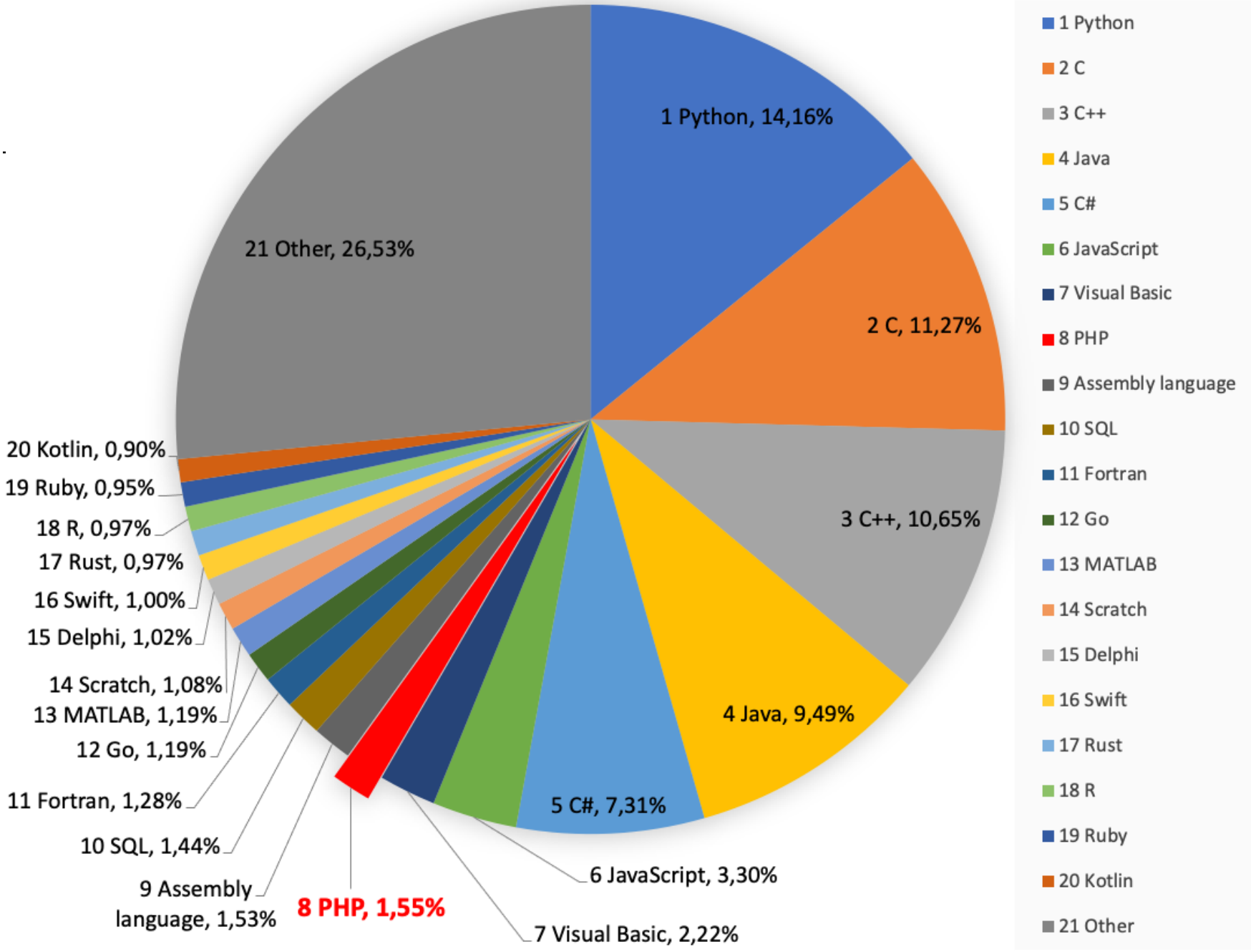
Chapitre 2 : le langage PHP-8, types, opérateur et structures de base

PHP (Hypertext Preprocessor)

- Langage de script généraliste et Open Source :
  - Tire son origine de PHP/FI (1995, Rasmus Lerdorf) "Personal Home Page"
  - PHP 3.0 (1998, Zeev Suraski et Andi Gutmans)
    - "People Hate Perl" (non officiel)
  - PHP 4.0 (2000), début des aspects objets
  - PHP 5.0 (2004), modèle objet complet
    - "PHP : Hypertext Preprocessor"
  - PHP 7.0 (2016), amélioration vitesse et contraintes
    - Contrainte sur la valeur de retour des fonctions
    - Opérateurs <=> et ??
  - PHP 8.0 (2020), arguments nommés, plus de typage, match, …
- Spécialement conçu pour le développement d'applications web
  - prévu d'emblée pour être intégré directement dans les pages HTML (ou dans des documents XML)
  - permet la génération dynamique des pages HTML

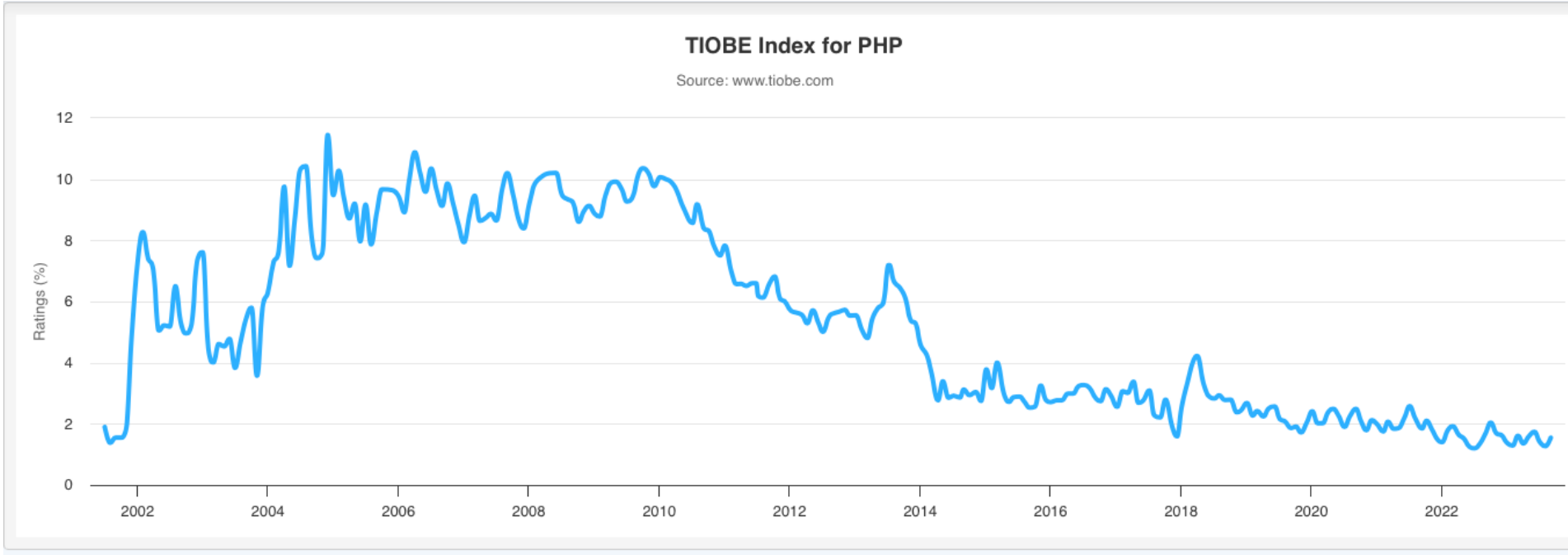


Source : **TIOBE Index Sept 2020**



Source : **TIOBE Index Sept 2023**

Evolution popularité PHP



Stable depuis 2014, mais avec une légère et constante réduction

Source : **TIOBE Index Sept 2023**

PHP

- Le serveur web **peut être configuré** pour déclencher **l'interpréteur PHP** aux pages portant l'extension .php
- Le serveur Web lance l'interpréteur PHP qui exécute le script PHP
- Le résultat de l'exécution est renvoyé au client
- Installation de PHP avec un serveur Web :
  - Installation de PHP en tant que **CGI**
    - PHP est installé de façon autonome => à chaque fois que le serveur web à besoin de traiter du code PHP, il y a lancement et arrêt d'un processus
  - Installation de PHP sous forme de **module** Apache (meilleures performances)
    - PHP est intégré au serveur web => le code PHP est exécuté directement par le processus du serveur web
- PHP sans serveur WEB : comme tous les autres interpréteurs
  - ex: shell, perl, => lancer l'interpréteur sur le fichier à interpréter

PHP – le langage

- PHP 7 = Langage Orienté Objet (LOO) - modèle objet à simple héritage complet (attributs et méthodes)
- Modèle objets style "Génie Logiciel" (vs Données, IA, Interfaces)
  - A base de classes et d'instances
- Un Langage de programmation à part entière
  - utilisable en **dehors du contexte** Web
  - Peut produire **n'importe quel type de données** en sortie standard
  - Gestion des exceptions et les contrôles de type (sûreté de programmation)
  - Un SGBDR intégré: SQLite
  - Existence d'un socle commun aux SGBD : PDO (PHP Data Object)
- Une Syntaxe « à la C », un extension objet inspiré de C++ et Java
- Faiblement typé, 4 types : booléen, entier, flottant, chaîne,
- Typage dynamique : la variable prend le type de la valeur affectée

Syntaxe de base

```
<?php
    // Premier programme PHP
    print("Hello world !\n");
?>

ou

<?= "Hello world !\n" ?>
```

- délimité par les balises d'échappement (début et fin)
  - <?php … ?>
- instructions terminées par un point-virgule ;
- bloc d'instructions délimité par des accolades { }
- des commentaires :
  - // ou # commentaire sur une ligne
  - /*… */ commentaire sur plusieurs lignes

Balises courtes, échappement conditionnel

```
<?= "Hello world !\n" ?>
La valeur de $A est <?= $A ?> !

<?php if ($expression == true):    ?>
  Ceci sera affiché si l'expression est vrai.
<?php else: ?>
  Sinon, ceci sera affiché.
<?php endif;    ?>
```

- Un raccourcis de <?php echo ... : <?=
- **Tout ce qui est en dehors de balises est produit en sortie sans modification**
- Balise conditionnelle : seule exception à la règle ci dessus sur la sortie hors des balises.

Les variables

```
<?php
    print($A); // ERREUR !
?>
```

- Démarrent avec le symbole $ (cf. le shell)
- Non déclarés, mais doivent être **initialisées** avant lecture
- Attention à la casse : $x != $X
- Test existence d'une variable : **isset**()

Les types des variables

```
#!/usr/bin/php
<?php
    $A = 0;             // Entier
    $B = 'TEST';        // Chaine
    $C = 0.567654;      // Flottant
    $D = TRUE;          // Booléen
    print($A);print($B);print($C);print($D);
?>
```

- 4 types simples : booléen, entier, réel (flottant), chaîne de caractères
- Variables :
  - Non déclarés, non typées, mais doivent être initialisée avant lecture
  - Typage dynamique : peut changer en cours d'utilisation.

Affichage : 0TEST0.5676541

Les constantes

```php
<?php
    const DEG=360; // Une déclaration (après PHP 5.3)
    define("PI",3.141592653589793); // Une instruction

    $A = 2 * PI / DEG;

?>
```

- Les constantes : define et const
- pas de $, par convention => en majuscule
    - define("TAUX",6.55957);
- Définition UNIQUE des constantes
  - la valeur et le type sont définitifs.
  - 'define' est une instruction et peut apparaître dans la branche d'un 'if'
  - 'const' doit être déclaré de manière globale (et est sensible à la casse)

Dynamicité des types

```
<?php
    $A = '2';    var_dump($A); // $A est de type chaine
    $A += 3;     var_dump($A); // $A est de type int
    $A += 1.0;   var_dump($A); // $A est de type float
?>
```

- Choisit le type numérique adéquat pour représenter la valeur
- Affiche :

string(1) "2"

int(5)

float(4)

Dynamicité des types : conversion des chaines

```
<?php
    $A = 1 + '10';   var_dump($A); // type int
    $A = 1 + '10.0';  var_dump($A); // type float
    $A = 1 + '-5.4E4';  var_dump($A); // type float
    $A = 1 + "20 degrés ce matin"; var_dump($A); // type int
    $A = 1 + "il fait 20 degrés ce matin"; var_dump($A); //int
?>
```

- Transformation automatique d'une chaine dans un contexte numérique.
- La chaine doit commencer par une valeur numérique valide
- Si la présence de '.', 'E' ou 'e' alors c'est un float
- Sinon c'est un int
- Dans les autres cas la chaine vaut 0 en PHP 5 mais provoque un **warning** en PHP 7, c'est une erreur en PHP 8

Dynamicité des types : conversion des chaines

```
<?php
    $A = 1 + '10';   var_dump($A); // type int
    $A = 1 + '10.0';  var_dump($A); // type float
    $A = 1 + '-5.4E4';  var_dump($A); // type float
    $A = 1 + "20 degrés ce matin"; var_dump($A); // type int
    $A = 1 + "il fait 20 degrés ce matin"; var_dump($A); //int
?>
```

- Résultat :

int(11)

float(11)

float(-53999)

int(21)

int(1) et **warning (et erreur)**

Casting de type

- Comme en C, un casting permet de changer le type d'une expression
- Expression
  - 1 constante
  - 1 variable
  - 1 fonction
  - toute combinaison avec les opérateurs
- (int), (integer) : entier
- (bool), (boolean) : booléen
- (float), (double), (real) : flottant
- (string) : chaine
- (array) : tableau
- (object) : objet
- (unset) : devient NULL
- (b), (binary) : motif binaire

Casting de type

```
<?php
    $A =  (int) '10';   var_dump($A);
    $A =  (float)'10';  var_dump($A);
    $A =  (string) (10 + 1.5);  var_dump($A);
    $A = (unset) "exemple";
    $A = (array) "ABCD"; var_dump($A);
?>
```

```
int(10)
float(10)
string(4) "11.5"
array(1) {
  [0]=>
  string(4) "ABCD"
}
```

Chaine de caractères

- **Différent** du standard C
- Un tableau de caractères avec un entier qui indique la taille
- Caractère de code **ASCII 0 autorisé**
- Utilisé pour implanter le type "Byte" (octet)
  - la plus petite unité adressable d'un ordinateur
  - actuellement de type octet
- Attention : en fonction du codage, un caractère peut être codé par plus d'un octet.
  - Certaine fonctions travaillent sur l'octet : substr(), strpos(), strlen() et strcmp()
  - D'autre travaillent en connaissant l'encodage des caractères : htmlentities()

Chaine avec caractères spéciaux

```
<?php
    $A =  13;   var_dump($A);
    $A = "La valeur est $A\n"; var_dump($A);
    $A =  22;
    $A = 'La valeur est $A\n'; var_dump($A);
?>
```

- Inspiré du shell et du langage C
- '.....' : une chaine littérale constante
- "....." : une chaine avec interprétation des caractères spéciaux
  - le $, les caractères ASCII \n, \t, \r, ...

Caractères spéciaux

- \n Fin de ligne (LF ou 0x0A (10) en ASCII)
- \r Retour à la ligne (CR ou 0x0D (13) en ASCII)
- \t Tabulation horizontale (HT or 0x09 (9) en ASCII)
- \v Tabulation verticale (VT ou 0x0B (11) en ASCII)
- \e échappement (ESC or 0x1B (27) en ASCII)
- \f Saut de page (FF ou 0x0C (12) en ASCII)
- \\ Antislash
- \$ Signe dollar
- \" Guillemet double
- \[0-7]{1,3} La séquence de caractères correspondant à cette expression rationnelle est un caractère, en notation octale
- \x[0-9A-Fa-f]{1,2} La séquence de caractères correspondant à cette expression rationnelle est un caractère, en notation hexadécimale

Traitement des chaînes

- Affichage simple : `echo() print()`

```
echo('Bonjour les étudiants');
print('Bonjour les étudiants');
echo 'Bonsoir les étudiants';
print 'Bonsoir les étudiants';
echo 'PHP',5,' est super ',$val;
```

- Accès à un caractère d'une chaîne

```
$chaine = "vive les pommes !";
echo $chaine[1]  //affiche i
```

- Affichage avec masque `printf() sprintf()`

```
$masque = 'la dernière version de %s est %d';
printf ($masque,'PHP', 5.3);
$chaine = sprintf ($masque,'PHP', 5.3);
```

Codage des chaines de caractères

```
<?php
    $A =  "é";    var_dump($A);
    print(strlen($A));
?>
```

string(2) "é"

2

Les booléens : conversion automatique

- Considéré comme FAUX :
  - false, FALSE
  - 0 et 0.0
  - '' et "0"
  - un tableau vide
  - un objet vide (sans attributs)
  - NULL
- Tout le reste est considéré comme VRAI !

Conversions en booléens

```
$A =  (bool) '';       var_dump($A);
$A =  (bool) 0;        var_dump($A);
$A =  (bool) 3+1;      var_dump($A);
$A =  (bool) (3+1);    var_dump($A);
$A =  (bool) (2-3+1); var_dump($A);
$A =  (bool) (2-3);    var_dump($A);
```

Conversions en booléens

```
$A =  (bool) '';       var_dump($A);
$A =  (bool) 0;        var_dump($A);
$A =  (bool) 3+1;      var_dump($A);
$A =  (bool) (3+1);    var_dump($A);
$A =  (bool) (2-3+1); var_dump($A);
$A =  (bool) (2-3);    var_dump($A);
```

bool(false)
bool(false)
int(2)
bool(true)
bool(false)
bool(true)

Conversions en booléens

```
$A =  (bool) "true";  var_dump($A);
$A =  (bool) 'true';  var_dump($A);
$A =  (bool) 'false'; var_dump($A);
$A =  (bool) '0 is not true';  var_dump($A);
$A =  (bool) '0';     var_dump($A);
$A =  (bool) '0.0';   var_dump($A);
```

Conversions en booléens

```
$A =  (bool) "true";  var_dump($A);
$A =  (bool) 'true';  var_dump($A);
$A =  (bool) 'false'; var_dump($A);
$A =  (bool) '0 is not true';  var_dump($A);
$A =  (bool) '0';     var_dump($A);
$A =  (bool) '0.0';   var_dump($A);
```

bool(true)
bool(true)
bool(true)
bool(true)
bool(false)
bool(true)

Opérateurs

- Opérateurs arithmétiques

  + - * / %

- Opérateurs de comparaison

  == != < <= > >= === !== <=>

- Opérateurs booléens

  ! AND && OR || XOR

- Opérateurs d'incrémentation, de décrémentation

  ++ -- += -=

- Opérateurs d'affectation, concaténation

  = . .=

Tableau associatifs

```
$tab = array('pomme','poire','pêche');
$tab[]= 'pomme'; $tab[]= 'poire'; $tab[]= 'pêche';
$tab = array (
'prenom'      => 'toto',
'age'         => 10,
'note'        => 0 );
echo "{$tab['prenom']} a eu la note{$tab['note' ]}";
```

- Comme les tableaux en C mais allocation automatique
- Ajout en fin de tableau : pas de valeur d'index
- L'index est un entier ou une chaine de caractère
- Instruction spéciale pour le parcours : foreach

Structures de contrôle

- if/ if .. else / if .. elseif …
- while / do .. while
- for, switch, ...

```
$i = 1;
while ($i <= 10) {
   print $i;$i++;}
```

```
switch ($val) {
 case  "pomme":
       $i=0;
       break;
 case  "poire":
       $i=1;
       break;
 default:
       $i=2;
       break;
}
```

```
for($i=1;$i<=10;$i++){
   print "$i";}
```

```
if ($i == 0) {
  print "i vaut 0";}
elseif ($i == 1) {
        print "i vaut 1";}
elseif ($i == 2) {
        print "i vaut 2";}
else {
  print "i différent";}
```