Chaînes de caractères

Chaîne de caractères ?

- **Série de caractères.**
- **1 caractère = 1 octet ➔ 256 valeurs différentes.**
  ➔Pas de support d'UNICODE.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STH	ETH	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	CD2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8	€	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	□	Ž	□
9	□	'	'	"	"	•	–	—	˜	™	š	›	œ	□	ž	Ÿ
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

codage "iso-8859-1"

➔ **Il est possible d'avoir des retours chariots, tabulations, etc dans une chaîne.**

francois.piat@univ-fcomte.fr

Guillemets

- **Guillemets simples**

```
$prenom = 'François';
```

- **Guillemets doubles**

```
$nom = "Piat";
$c = "Je m'appelle $prenom $nom";
```

➔ Je m'appelle François Piat

- **Saut de lignes, tabulations, etc.**

```
$c = "Je
m'appelle
      $prenom       $nom";
```

- **Protection de caractères**

```
'Je m\'appelle François'
"Je lui ai dit \"Bonjour\""
```

francois.piat@univ-fcomte.fr

heredoc

Syntaxe heredoc

```
<<<IDENTIFIANT
texte
IDENTIFIANT;
```

```
$txt = <<<TEXTE
Bonjour,
    Je m'appelle $prenom, et je suis enseignant
    à l'Université de Franche-Comté.
TEXTE;
```

francois.piat@univ-fcomte.fr

Concaténation

- Opérateur . (point)

```
$a = 'François';
$b = 'Piat';
$c = $a . $b;
```

➔ FrançoisPiat

```
$c = $a . ' ' . $b;
```

➔ François Piat

- Concaténation et affectation avec l'opérateur .=

```
$c .= ' enseigne PHP';
```

➔ François Piat enseigne PHP

- Eviter les concaténations
  - Utiliser la substitution de variables avec les guillemets doubles
  - Utiliser l'instruction echo avec des paramètres séparés par des virgules

francois.piat@univ-fcomte.fr

Afficher une chaîne

- Afficher une chaîne = envoyer la chaîne au navigateur
- Instruction echo

```
echo '<h1>', $titre, '</h1><p>', $txt, '</p>';
```

- Instruction print

```
print '<h1>'.$titre.'</h1><p>'.$txt.'</p>';
```

- Préférer d'entourer les chaînes avec des guillemets simples et d'utiliser l'instruction echo
- Affichage avec formatage : printf() et sprintf();

francois.piat@univ-fcomte.fr

Bibliothèque

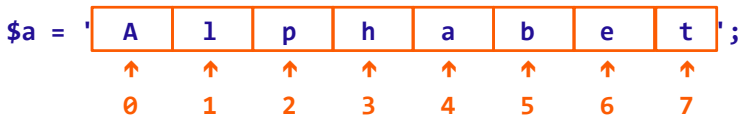
- Bibliothèque très complète pour gérer les chaînes de caractères (98 fonctions) mais
- Pas de normalisation dans les noms de fonction

```
substr()   strpos()   str_replace()   trim()
   htmlentities()   html_entity_decode()
```

- Pas de règles dans l'ordre de passage des paramètres
- Utilisation de l'aide de PHP obligatoire
  http://www.php.net/manual/fr/book.strings.php
- Si texte en UTF8 ➔ utiliser la bibliothéque `mb_string`
  http://www.php.net/manual/fr/ref.mbstring.php

francois.piat@univ-fcomte.fr

Fonctions incontournables

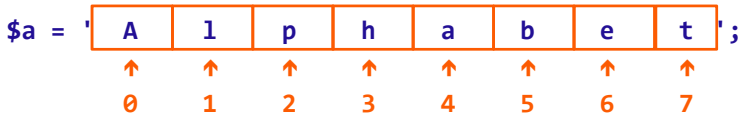


- `strlen()`
- `substr()`
- `strpos()`
- `trim()`
- `explode()`

```
strlen($a)          ➔ 8
substr($a, 1, 3)    ➔ lph
substr($a, 2)       ➔ phabet
substr($a, -2)      ➔ et
strpos($a, 'h')     ➔ 3
strpos($a, 'z')     ➔ FALSE
strpos($a, 'A')     ➔ 0
```

francois.piat@univ-fcomte.fr

Fonctions incontournables



`strpos($a, 'A')` ➔ 0

```
$a = 'Alphabet';
$pos = strpos($a, 'A');
if ($pos == FALSE) {
    echo 'A pas trouvé dans Alphabet';
} else {
    echo 'A est la lettre ', $pos , 'dans Alphabet';
}
```

`if ($pos === FALSE)`

Codage serveur ➔ client

Dans le sens PHP ➔ HTML

- Protéger les caractères de la syntaxe HTML
- Remplacer les caractères accentués

`htmlentities(chaîne, gestion_guillemets, encodage)`

< ➔ <
> ➔ >
é ➔ é
è ➔ è
à ➔ à
î ➔ î
...

```
$x = 'a < b => b > a';
htmlentities($x, ENT_QUOTES, 'ISO-8859-1');
```

➔ a > b => b < a

```
$x = 'François';
htmlentities($x, ENT_QUOTES, 'UTF8');
```

➔ François

Codage serveur ➔ client

Lutter contre les attaques XSS (cross site scripting).

```
htmlentities('<script>location="http://concurrent.com"</script>',
ENT_QUOTES, 'ISO-8859-1');
```

```
&lt;script&gt;location=&quot;http://concurrent.com&quot;&lt/script&gt;
```

Supprimer les tags HTML dans une chaîne : `strip_tags()`

```
strip_tags('<script>location="http://concurrent.com"</script>');
```

```
location="http://concurrent.com"
```

francois.piat@univ-fcomte.fr

Codage client ➔ BD

- **Dans le sens saisie utilisateur ➔ serveur base de données**
- **Protéger les caractères avec signification particulière**

`mysqli_real_escape_string()`

`mysqli_escape_string()`

`addslashes()`

francois.piat@univ-fcomte.fr

Tableaux

Tableaux ?

- http://php.net/manual/fr/language.types.array.php :

"Ce type est optimisé pour différentes utilisations ; il peut être considéré comme un tableau, une liste, une table de hashage, un dictionnaire, une collection, une pile, une file d'attente et probablement plus."

	1 entier	Tableau de 100 000 entiers
C	8 octets	0,76 Mo
PHP	48 octets	4,7 Mo + 9,3 Mo ➔14 Mo

francois.piat@univ-fcomte.fr

Tableaux ?

- **Struct** (records, tuples) : une valeur contenant d'autres valeurs.
- **Tableau** (array) : collection de taille fixe dont les éléments sont identifiés par un index numérique.
- **File** (queue) : collection ordonnée respectant l'ordre *1er arrivé, 1er sorti* (FIFO).
- **Pile** : collection ordonnée respectant l'ordre *dernier arrivé, 1er sorti* (LIFO).
- **Ensemble** (set) : collection sans ordre spécifique, adapté pour réaliser des opérations d'union, d'intersection, de complément.
- **Dictionnaire** (dictionary, map, table de hashage) : collection de couples clé/valeur où chaque clé est unique.

2 sortes de tableaux

- Tableaux avec un indice numérique :
  l'indice est un entier positif ou égal à 0

```
$t[0]    $t[1]    $t[100]
```

- Tableaux associatifs (table de hachage) :
  l'indice est une chaîne de caractères

```
$t['nom']    $t['prenom']    $t['tél']
```

- Nombre d'éléments dans un tableau : count()

```
$nb = count($t);
```

Contenu mixte

- **Les éléments d'un tableau peuvent être de n'importe quel type.**

```
$t[0] = 'Pierre';
$t[1] = 'Paul';
$t[2] = 'Jacques';
```

```
$t[0] = 15;
$t[1] = 10;
$t[2] = 18;
```

- **Des éléments de types différents peuvent se trouver dans le même tableau.**

```
$t[0] = 'Pierre';
$t[1] = 15;
$t[2] = true;
$t[3] = array('Etudiant', 100, false);
```


Création indice numérique

- `$var[] = valeur;`

```
$t[] = 10;
$t[] = 8;
$t[] = 15;
```

- `array()`

```
$t = array(10, 8, 15);
```

- `range()`

```
$t = range(100, 150);
$t = range('a', 'z');
```

- `explode()`

```
$t = explode('/', '25/08/2010');
```

- `array_fill()`

```
$t = array_fill(0, 10, true);
```

- `array_merge()` `array_diff()` `array_filter()`

Indice numérique


```
$ta[] = 'abc';
$ta[] = 10;
$ta[] = 1234;
$ta[] = -1;
```

0	abc
1	10
2	1234
3	-1

```
$tb[] = 'abc';
$tb[] = 10;
$tb[6] = 1234;
$tb[8] = -1;
```

0	abc
1	10
6	1234
8	-1

$tb a des trous dans ses indices :
2, 3, 4, 5 et 7
➔ Pas de boucle avec **for**

```
$tc[5] = 'abc';
$tc[] = 10;
$tc[] = 1234;
```

5	abc
6	10
7	1234

L'index de **$tc** ne commence pas à 0
➔ Boucle avec **for** hasardeuse

Création tableaux associatifs

- **tableau associatif = table de hashage = dictionnaire**
  ➔associer une valeur à nom

- `$var['nom_index'] = valeur;`

```
$t['nom'] = 'Pierre';
$t['prénom'] = 'Paul';
$t['redoublant'] = false;
$t['nbAbsence'] = 0;
$t['moyenne'] = 15.75;
```

Fonction array()

- Sans argument pour créer un tableau vide

```
$t = array();
```

```
$ta = $tb = $tc = array();
```

- Avec argument pour créer un tableau non vide

```
$t_num = array('Pierre', false, 10, 14, 'abs');
```

```
$t_asso = array('nom' => 'Pierre',
                'absent' => false,
                'note1' => 10,
                'note2' => 14,
                'note3' => 'a
```

```
$t_mieux = array('nom' => 'Pi
                'absent' => f
                'notes' => array(15, 16, 'abs'));
```

```
$t_mieux['notes'][0] ➔ 10
$t_mieux['notes'][1] ➔ 14
$t_mieux['notes'][2] ➔ abs
```

francois.piat@univ-fcomte.fr

Lecture de tableaux

- Directement quand on connaît l'indice de façon certaine

```
$a = $t[5];
$b = $t['nom'];
```

- En vérifiant si l'indice est présent

```
if (isset($t[6])) {
    $c = $t[6];
} else {
    $c = null;
}
```

```
$c = (isset($t[6])) ? $t[6] : null;
```

- Par itérations : `for` et `foreach` ou fonctions spécifiques

francois.piat@univ-fcomte.fr

Boucle for (1)

- **for est à utiliser**
  - avec un tableau à indice numérique
  - si il n'y a pas de trous dans les indice.

```
$t[] = 'abc';
$t[] = 10;
$t[6] = 1234;
$t[8] = -1;
```

0	abc
1	10
6	1234
8	-1

```
$nb = count($t);   4
for ($i = 0; $i < $nb; $i++) {
    echo $t[$i];
}
```

$i	$t[$i]
0	abc
1	10
2	???
3	???

francois.piat@univ-fcomte.fr

Boucle for (2)

Ne jamais appeler une fonction dans le test d'itération.

```
for ($i = 0; $i < count($t); $i++) {
     echo $t[$i];
}
```

```
$nb = count($t);
for ($i = 0; $i < $nb; $i++) {
      echo $t[$i];
}
```

```
for ($i = 0, $nb = count($t); $i < $nb; $i++) {
      echo $t[$i];
}
```

francois.piat@univ-fcomte.fr

Boucle for (3)

A utiliser quand l'ordre de traitement n'a pas d'importance :

```
for ($i = count($t) - 1; $i >= 0; $i--) {
    ...
}
```

```
for ($i = count($t); $i--; ) {
    ...
}
```


Boucle foreach (1)

- `foreach` est à utiliser
  - avec un tableau associatif
  - avec un tableau à indice numérique, dont les indices ne sont pas consécutifs
- `foreach` travaille sur une copie du tableau si on modifie le tableau au cours de la boucle.
  Attention à l'occupation mémoire avec des tableaux importants

Boucle foreach (2)


```
foreach ($Tableau as $Valeur) {
    ...
}
```

```
$t['nom'] = 'Pierre';
$t['présent'] = true;
$t['moyenne'] = 15.36;
```

```
foreach ($t as $val) {
    echo $val;
}
```

$val
Pierre
true
15.36

```
$t[0] = 125;
$t[5] = 523;
$t[10] = 4587;
```

```
foreach ($t as $val) {
    echo $val;
}
```

$val
125
523
4587

francois.piat@univ-fcomte.fr

Boucle foreach (3)


```
foreach ($Tableau as $Clé => $Valeur) {
    ...
}
```

```
$t['nom'] = 'Pierre';
$t['présent'] = true;
$t['moyenne'] = 15.36;
```

```
foreach ($t as $cle => $val) {
    echo $cle, $val;
}
```

$cle	$val
nom	Pierre
présent	true
moyenne	15.36

francois.piat@univ-fcomte.fr

Fonctions d'itération (1)

`current()`	Retourne la valeur de l'élément courant
`reset()`	Place le pointeur sur le premier élément et en retourne la valeur
`end()`	Place le pointeur sur le dernier élément et en retourne la valeur
`next()`	Place le pointeur sur l'élément suivant et en retourne la valeur
`prev()`	Place le pointeur sur l'élément précédent et en retourne la valeur
`each()`	Retourne la clé et la valeur de l'élément courant (sous la forme d'un tableau), et place le pointeur sur l'élément suivant
`key()`	Retourne la clé de l'élément courant

francois.piat@univ-fcomte.fr

Fonctions d'itération (2)


```
$t['nom'] = 'Pierre';
$t['présent'] = true;
$t['moyenne'] = 15.36;
```

```
reset($t);
while ($val = current($t)) {
    echo $val;
    next($t);
}
```

$val
Pierre
true
15.36
false

```
$t['nom'] = 'Paul';
$t['présent'] = false;
$t['moyenne'] = 0;
```

➔ s'arrête à **`$t[présent']`**

$val
Paul
false

```
reset($t);
while ($elt = each($t)) {
    echo $elt[0], $elt[1];
}
```

$elt[0]	$elt[1]
nom	Paul
présent	false
moyenne	0

francois.piat@univ-fcomte.fr

Autres fonctions (1)

- **Une bibliothéque de fonctions très fournie (77 fonctions)**
  - Vérifier dans l'aide de PHP les fonctions
- **Vérification de clés et/ou de valeurs**

```
array_keys()      array_values()      array_key_exists()
in_array()        array_search()
```

- **Tris**
  - Par valeur avec changement d'indice : `sort()` `rsort()` `usort()`
  - Par valeur : `asort()` `arsort()` `uasort()`
  - Par clés : `ksort()` `krsort()` `ukasort()`
  - Tous les tris sont faits sur le tableau original

Autres fonctions (2)

- **Opérations ensemblistes sur des tableaux**

```
array_diff()      array_intersect()
array_merge()     array_unique()
```

- **Supprimer, insérer, remplacer des séries d'éléments**

```
array_splice()    array_slice()
```

- **Piles**

```
array_push()      array_pop()
array_shift()     array_unshift()
```

Autres fonctions (3)

Transformer les éléments d'un tableau en variables

list()

```
$a = '31/12/1999';
$t = explode('/', $a);
list($J, $M, $A) = $t;
```

```
$t[0] = 31
$t[1] = 12
$t[2] = 1999
```

```
$J = 31
$M = 12
$A = 1999
```

extract()

```
$tab['Pierre'] = 18;
$tab['Paul'] = 10;
$tab['Jacques'] = 15;
extract($tab);
```

```
$Pierre = 18
$Paul = 10
$Jacques = 15
```

francois.piat@univ-fcomte.fr