Coding in PHP with only 6 characters

lendable

We're hiring!

Benoit VIGUIER
@b_viguier

FORUMPHP
PARIS2024


LO
VE

Constraints lead to creativity

Brainfuck

1993 - Urban Müller

```
+-<>[],.
```

Hello World

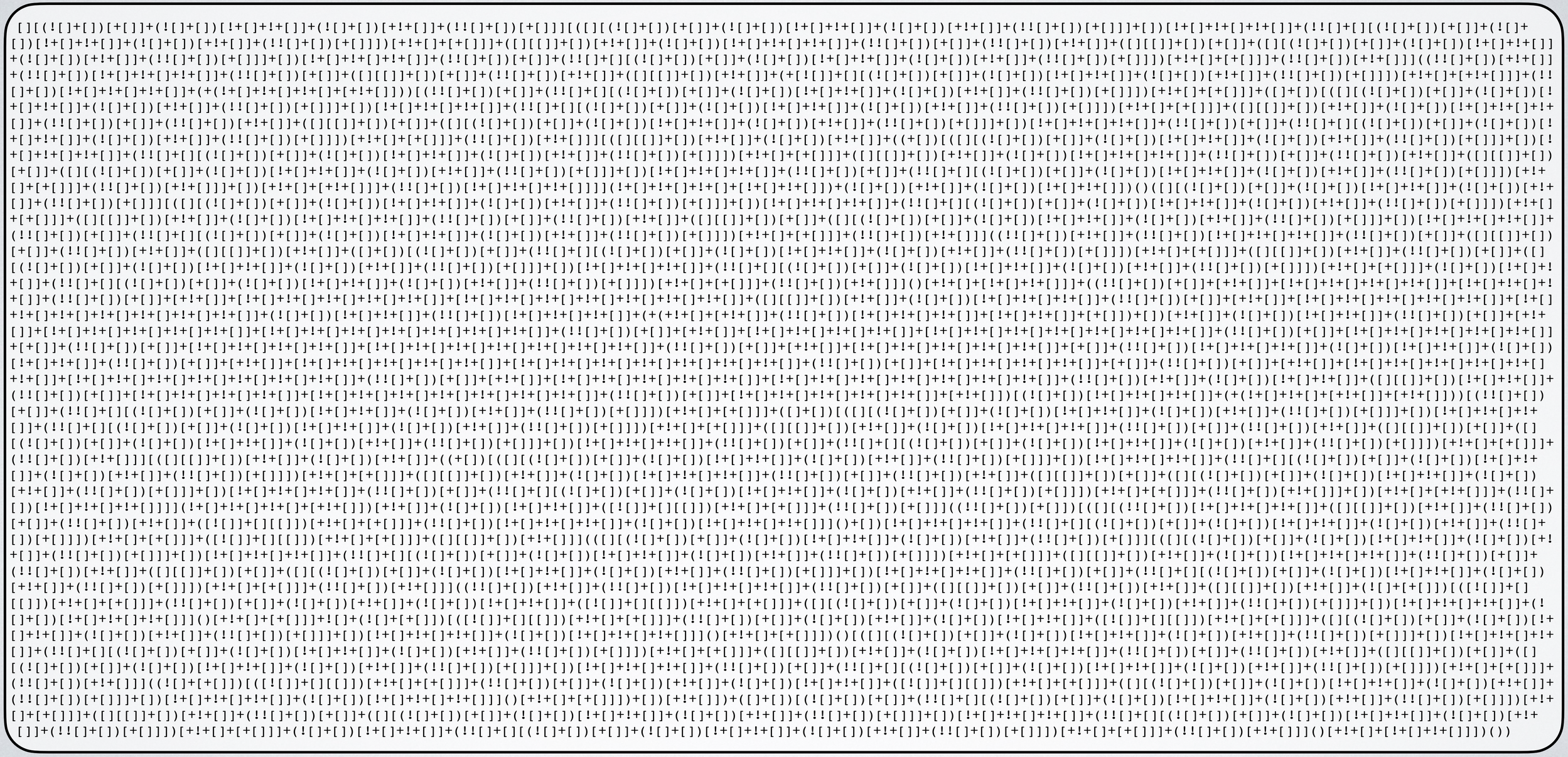
```
++++++++[>++++[>++>+++>++
+>+<<<<-]>+>+>->>+
[<]<-]>>.>---.+++++++..++
+.>>.<-.<.++
+.------.--------.>>+.>++.
```

JsFuck

2010 - Internet

```
[]()!+
```

Hello World



What about PHP???

splitline/PHPFuck

7 characters, Php 7, Python script

```
([+.^])
```

lebr0nli/PHPFun

6 characters, Php 7, Python script

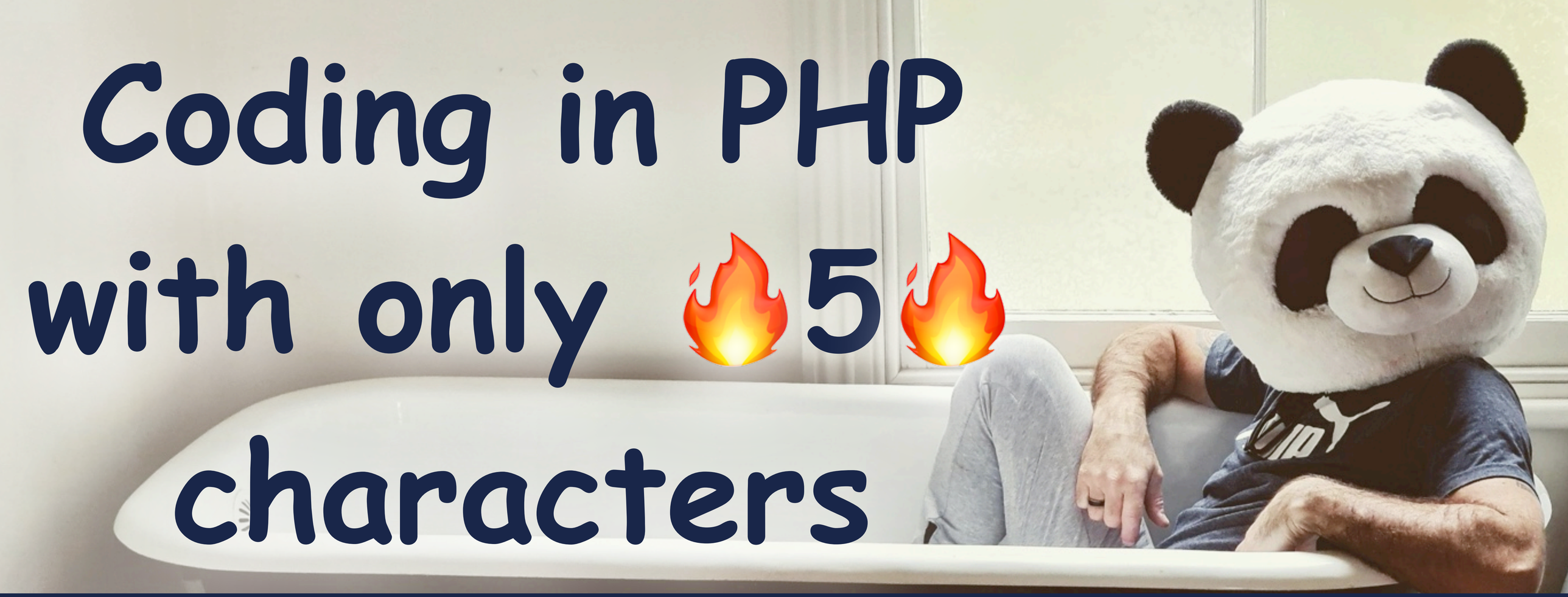
```
([•^])
```

arxenix/phpfuck

5 characters, Php 7, Python script

```
(^.9)
```

Coding in PHP with only 🔥5🔥 characters

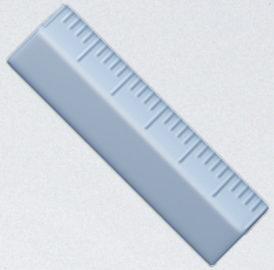


lendable

We're hiring!

Benoit VIGUIER
@b_viguier

FORUMPHP PARIS2024

Rules 

- Must work **out of the box** (no extensions, packages…)
- **No warning** (cannot disable them)
- **Blank characters** are counted (space, tab, line-break…)
- PHP tags are not part of the challenge (**<?php … ?>**)

The big picture!

Usual PHP ➡️ 13 (and more)

```php
echo "Hello World";
```

```
echo "HloWrd;
```

eval ➡️ 18 (and more)

```php
eval('echo "Hello World";');
```

```
echo "HloWrd;vl()'
```

Creating strings with few characters

🤔

⭐

chr

```
chr(int $codepoint): string
```

Generate a single-byte string from a number

Creating all possible strings

```
chr(101).chr(99).chr(104).chr(111)
```

```
chr(0123456789).
```

21 characters max!

```
eval(chr(101).chr(99).chr(104).chr(11
1).chr(32).chr(34).chr(72).chr(101).ch
r(108).chr(108).chr(111).chr(32).chr(8
7).chr(111).chr(114).chr(108).chr(100)
                .chr(34).chr(59));
```

```
eval(chr0123456789).;
```

Easy improvements

Creating all possible strings

```
chr(1+1+1+1+1+1+1+1+1+1)
```

```
chr(1+)
```

chr

```
chr(256) === chr(0)
```

Values outside the valid range (0..255) will be bitwise and'ed with 255

13 characters!

```
eval(chr(1+1+…1+1+1).chr(1+1+…+1+1+1)…);
```

```
eval(chr1+).;
```

12 characters!!!

```
<?php
  eval(chr(1+1+…1+1).chr(1+1+…+1+1)…)
?>
```

```
eval(chr1+).
```

Variable functions

chr

```
('chr')(101)
```

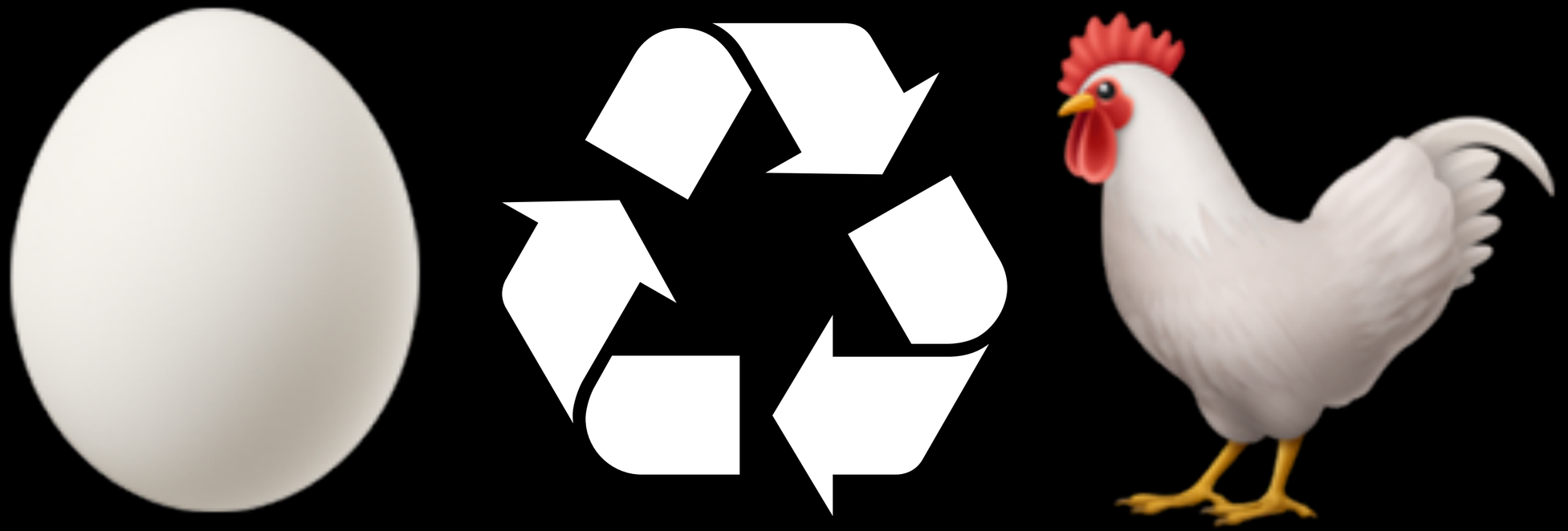
Generate a single-byte string from a number

Type juggling

chr

```
('chr')('101')
```

Generate a single-byte string from a number



^ XOR ^

^ (xor)

```
0b00001111  // 15
          ^
0b00110011  // 51
        ===
0b00111100  // 60
          ;
```

^ (xor)

```
('.'^'('^'e') === 'c';
('l'^'a'^'e') === 'h';
('a'^'v'^'e') === 'r';
```

^ (xor)

```
('.la'^'(av'^'eee'          )
          === 'chr';

('.la'^'(av'^'eeeeeeeee')
          === 'chr';
```

^ (xor)

```
('.'^'l'^'a'^'v'^'e')       === '0';
(')'^'^'^'.'^'l'^'a'^'e') === '1';
('^'^'l')                   === '2';
(')'^'l'^'v')               === '3';
('.'^'l'^'v')               === '4';
(')'^'^'^'.'^'l')           === '5';
('^'^'l'^'a'^'e')           === '6';
(')'^'l'^'a'^'v'^'e')       === '7';
(')'^'^'^'.'^'a')           === '8';
('.'^'a'^'v')               === '9';
```

9 characters

```
<?php
  eval((((('.'^'('^'e')…)))))
?>
```

```
eval('^.)
```

What about « eval »?

It’s a language construct 😕

More tricks!

🧙‍♀️

⭐⭐⭐

Type juggling with 9

```
(9).(9) === '99';
(9 ^ 9) === 0;
(0).(9) === '09';
('99' ^ 0) === 99;
(99 ^ 9) === 106;
(106).(9) === '1069';
('09'^'1069'^'99') === '80';
```

Infinity's trick 🤯

```
INF ===
9999999999999999999999999999999999999999
9999999999999999999999999999999999999999
9999999999999999999999999999999999999999
9999999999999999999999999999999999999999
9999999999999999999999999999999999999999
9999999999999999999999999999999999999999
9999999999999999999999999999999999999999
9999999999999999999999999999999999999999
  99999999999999999999999999999999999;
```

Infinity's trick 🤯

```
INF ===
999999999999999999999999999999999999999
999999999999999999999999999999999999999
999999999999999999999999999999999999999
999999999999999999999999999999999999999
999999999999999999999999999999999999999
999999999999999999999999999999999999999
999999999999999999999999999999999999999
999999999999999999999999999999999999;
```

309 '9's 😱

Infinity's trick 🤯

```
INF.(9)

====

'INF9';
```

Chr is back

```
('INF9' ^ '864' ^ "20\0")
===
'CHr';
```

Case insensitive

```
chr(42) === CHr(42)
```

(still) 9 characters

```
<?php
  eval('<str>')
?>
```

```
eval(9^.)
```

FFI

🤓

⭐⭐⭐⭐

Example

```
$ffi = FFI::cdef(
    // this is a regular C declaration
    "int printf(const char *format, ...);",
    "libc.so.6");


// call C's printf()
$ffi->printf("Hello %s!\n", "world");
```

Works with Zend functions

zend_eval_string

```
zend_result zend_eval_string(
        const char *str,
        zval *retval_ptr,
        const char *string_name
);
```

zend_eval_string

```
char                zend_eval_string(
                        char *str,
        int             retval_ptr,
                        char *string_name
);
```

eval

```
FFI::cdef(
    'char zend_eval_string(char*,int,char*);'
)->zend_eval_string(
    'echo "Hello World";',
    '0',
    ''
);
```

Variable functions

```
'FFI::cdef'(
    'char zend_eval_string(char*,int,char*);'
)->{'zend_eval_string'}(
    'echo "Hello World";',
    '0',
    ''
);
```

Callable array

```
['FFI::cdef'(
    'char zend_eval_string(char*,int,char*);'
    ),
    'zend_eval_string'
](
    'echo "Hello World";', '0', ''
);
```

8 characters!

```
<?php
   [<str>(<str>),<str>](<str>,<str>,<str>)
?>
```

```
[(,^.9)]
```

How to create arrays from strings?

🙃

⭐⭐⭐⭐⭐⭐

Starting point

```
[
    FFI::cdef(
        "char zend_eval_string(const char *, int, const char *);"
    ),
    "zend_eval_string"
]("echo 'Hello World';", 0, "");
```

For readability…

```
[
      FFI::cdef(
         "<c_def>"
      ),
      "<c_eval>"
]("<code>", 0, "");
```

...

```php
function foo(string $a, int $b): void {}
foo('Hello', 1);
foo(...['Hello', 1]);
```

Array unpacking, spread operator (Php 7.4)

Everything is array

```
[
        FFI::cdef("<c_def>"),
        "<c_eval>"
]("<code>", 0, "");
```

Everything is array

```
[
        FFI::cdef("<c_def>"),
        "<c_eval>"
](...["<code>", 0, ""]);
```

Json-strings are arrays

```
[
        FFI::cdef("<c_def>"),
        "<c_eval>"
](...json_decode(
        '["<code>", 0, ""]'
));
```

strings are functions

```
[
    'FFI::cdef'("<c_def>"),
     "<c_eval>"
](...'json_decode'(
     '["<code>", 0, ""]'
));
```

Readibility…

```
[
    'FFI::cdef'("<c_def>"),
     "<c_eval>"
](…);
```

Merging arrays

```
array_merge(
        ['FFI::cdef'("<c_def>")],
        ["<c_eval>"]
)(…);
```

array_map to call function

```
array_merge(
      array_map(
         'FFI::cdef',["<c_def>"]
      ),
      ["<c_eval>"]
)(…);
```

array_map to call function

```
array_merge(
      array_map(
         'FFI::cdef',["<c_def>"]
      ),
      ["<c_eval>"]
)(…);
```

array_map all the things

```
array_merge(
    array_map(
        'FFI::cdef',["<c_def>"]
    ),
    array_map(
        "strval",["<c_eval>"]
    )
)(…);
```

array_map

```
array_map(
        'foo', [0,1,2], [0,1,2]
) ===
[
        foo(0, 0),
        foo(1, 1),
        foo(2, 2),
];
```

array_map **ALL** the things

```
array_merge(
        array_map(
            'FFI::cdef',["<c_def>"]
        ),
        array_map(
            "strval",["<c_eval>"]
        )
)(…);
```

array_map **ALL** the things

```
array_merge(...array_map(

              "array_map",
            ["FFI::cdef", "strval"],
          [["<c_def>"],["<c_eval>"]]

))(…);
```

Strings everywhere

```
array_merge(...array_map(
      ...json_decode(
          '["array_map",
           ["FFI::cdef", "strval"],
          [["<c_def>"],["<c_eval>"]]]'
      )
))(…);
```

5 characters 😵

```
<?php
   'merge'(...'map'(...'json'('<ffi>')))
   (...'json'('["<code>",0,""]'));
?>
```

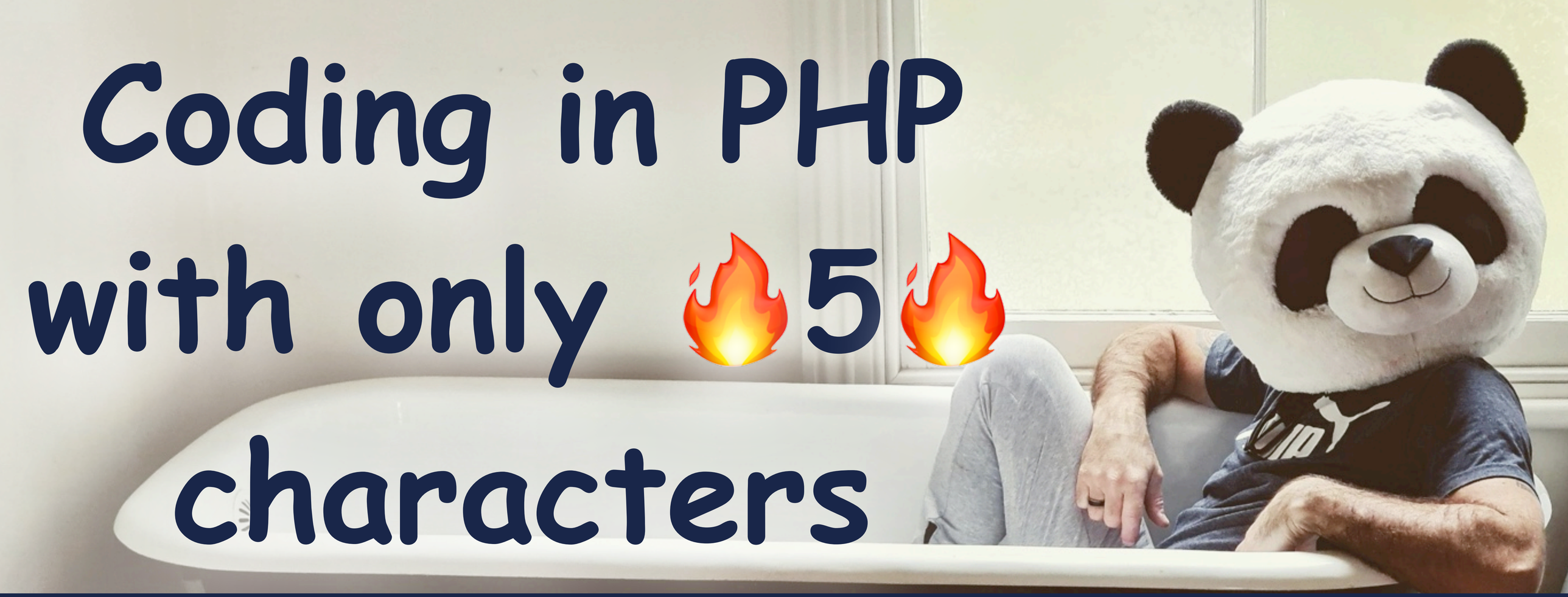
(^.9)

b-viguier/PhpFk

5 characters, Php **8.0**, **Php** script

```
(^.9)
```

Coding in PHP with only 🔥5🔥 characters



lendable

We're hiring!

Benoit VIGUIER
@b_viguier

FORUMPHP PARIS2024