

# MÓDULO

# PROGRAMAÇÃO PHP

UNIDADE

FUNÇÕES PHP



## ÍNDICE

---

OBJETIVOS.....	3
INTRODUÇÃO.....	4
1. CRIAR FUNÇÕES.....	5
2. PASSAR PARÂMETROS.....	8
3. VALORES DEVOLVIDOS.....	14
4. FUNÇÕES VARIÁVEIS.....	16
5. FUNÇÕES DO SISTEMA.....	18
CONCLUSÃO.....	27
AUTOAVALIAÇÃO.....	29
SOLUÇÕES.....	33
PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO.....	34
BIBLIOGRAFIA.....	35



## OBJETIVOS

---

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Criar funções em PHP.
- Reconhecer os tipos de dados que pode enviar para uma função.
- Devolver dados com uma função.
- Reconhecer as principais funções do PHP.

## INTRODUÇÃO

---

O código linear é um sistema de programação que não é muito benéfico na programação web, por isso as linguagens de programação implementam funções, que são conjuntos de código que podem ser chamados de qualquer lugar da página.

É o recurso mais utilizado na programação e consegue limpar, e estruturar o código.

# 1. CRIAR FUNÇÕES

---

Uma função é um excerto de código definido entre chavetas num identificador, que pode ou não receber parâmetros. As funções são utilizadas para criar uma estrutura de programa, limpar e evitar a repetição do código.

Para criar uma função é utilizada a palavra reservada `function` da linguagem PHP, um identificador e os parâmetros (se necessário) entre parênteses.

```
function funcao(argumentos...) { }
```

Dentro da função, qualquer tipo de código PHP pode ser executado e, no final, pode devolver um valor para um controlador que irá chamar a função usando o método `return`. Para que o valor seja obtido, deverá chamar a função atribuindo-o à variável que pretende que tenha o valor.

```
function soma($a, $b) {  
    return $a+$b;  
}  
  
$resultado = soma(4,5);  
echo '4 + 5 é igual a '.$resultado;
```

É importante referir que o PHP diferencia maiúsculas de minúsculas nos nomes das funções.

Por definição, todas as funções existem desde o início do programa. A exceção é a criação de funções condicionais, ou seja, a criação de uma condição na definição da função que evite que ela seja definida até que a execução passe por ela.

```
$colocarCor = true;
mostrarAzul();

/*aqui mostrará azul, enquanto, mesmo sendo criado mais abaixo, se
colocar mostrar Verde, dará erro */
if ($realizar) {
    function mostrarVerde(){
        echo "Verde.\n";
    }
}
if ($colocarCor) mostrarVerde();
function mostrarAzul(){
    echo "Azul.\n";
}
```

Também existe a possibilidade de criar funções dentro de funções, que podem ser úteis para controlar certos tipos de execuções condicionadas nas anteriores, uma vez que a função B contida na função A não pode ser invocada até que a função A seja invocada.

```
function A(){
    function B(){
        echo "funcionou";
    }
}
A();
B();

/* se trocar a ordem de chamada, dará erro */
```

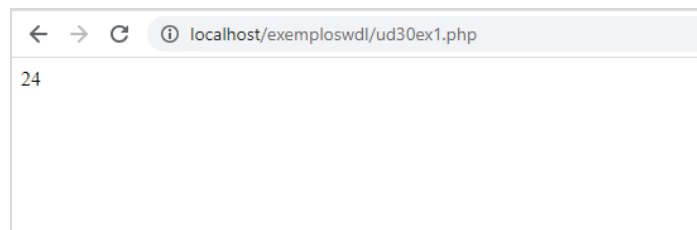
Por fim, uma função pode chamar-se a si própria. Para isso basta colocar a chamada dentro da própria função.



Neste exemplo, irá criar-se uma função para calcular o fatorial de um determinado número, neste caso 4. O fatorial de 4 é o resultado da multiplicação de todos os valores de 1 a 4 pelo resultado anterior, ou seja  $((1 * 1) * 2) * 3) * 4$ . O resultado final será 24.

```
<html>
<body>
<p>
<?php
    function fatorial($numero){
        if($numero==0) return 1;
        return ($numero*fatorial($numero-1));
    }
    echo fatorial(4);
?>
</p>
</body>
</html>
```

A visualização será:



## 2. PASSAR PARÂMETROS

---

Qualquer função pode receber quantos parâmetros pretender na forma de uma lista separada por vírgulas. O PHP suporta parâmetros passados por referência, por valor e qualquer tipo de dados como um vetor.

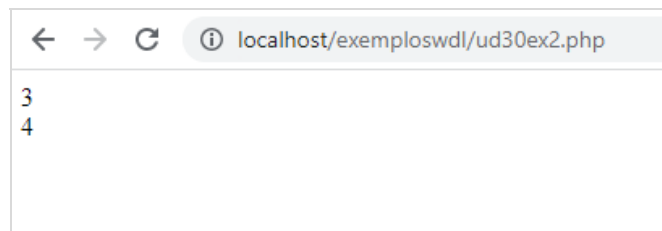
Passar um parâmetro por valor significa passar o valor (ou variável) diretamente para a função, para esta o manipular internamente, mas sem modificar o valor real. Se pretender modificar o valor, terá de utilizar o return para recolher o valor devolvido. Por exemplo:

```
<html>
<body>
<p>
<?php
    function teste1($valor){
        $valor ++;
        return $valor;
    }
    $teste = 3;
    $testeD = teste1($teste);
    echo $teste.'<br/>';
    echo $testeD;
?>
```

```
</p>
</body>
</html>
```

Ao executar este exemplo, verá o resultado: o primeiro valor mostrado é 3 e o segundo valor é 4.

A visualização será:



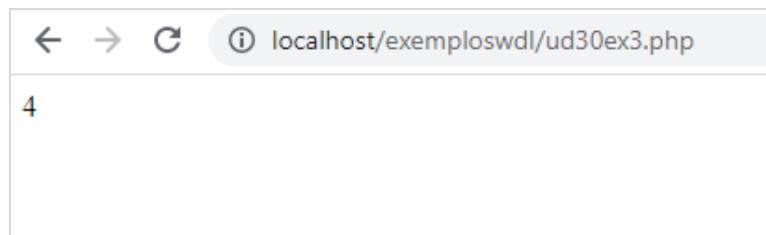
Ao passar o valor por referência significa que, se o valor é modificado dentro da função, também é modificado fora. Para passar um valor por referência, basta colocar o sinal “e” comercial (&) antes do parâmetro. Segue-se o exemplo anterior com esta mudança:

```
<html>
<body>
<p>
<?php
    function teste1(&$valor){
        $valor ++;
    }
    $teste = 3;
    teste1($teste);
    echo $teste.'<br/>';
?>
</p>
</body>
```

```
</html>
```

Ao executar o exemplo, verá que o valor devolvido é 4.

A visualização será:

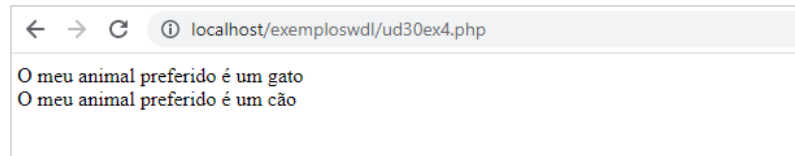


Outra opção que o PHP permite é o uso de parâmetros padrão, ou seja, um parâmetro que, se não receber nada, assume um valor padrão.

```
<html>
<body>
<p>
  <?php
    function animalPreferido($animal = "cão"){
      return 'O meu animal preferido é um '.$animal;
    }
    echo animalPreferido('gato')."<br>";
    echo animalPreferido();
  ?>
</p>
</body>
</html>
```

Ao executar este código, verá que na primeira vez mostrará "O meu animal preferido é um gato", enquanto na segunda vez, por não passar um parâmetro, mas ter um valor padrão, mostrará "O meu animal preferido é um cão".

A visualização será:



Uma função em PHP não precisa de ter parâmetros, mas se não tiver parâmetros e se os passar pela função, não originará um erro como aconteceria nas outras linguagens. Isso acontece porque as funções em PHP suportam listas de parâmetros, ou seja, não é necessário definir os parâmetros que irá receber.

Uma vez dentro da função PHP, existem duas funções PHP que permitem saber quantos parâmetros foram recebidos (`func_num_args()`) e obter o valor de qualquer um deles (`func_get_arg(n)`).

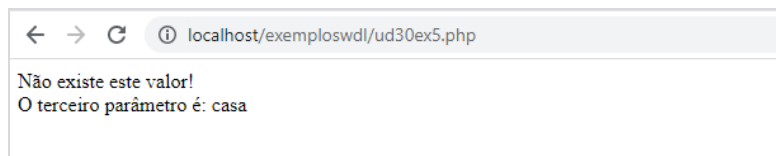
Neste exemplo, irá criar uma função que recebe `n` parâmetros e que indica o terceiro valor, desde que esse valor exista.

```
<html>
<body>
<p>
<?php
    function teste1(){
        $argumentos = func_num_args();
        if ($argumentos >= 3) {
            echo "O terceiro parâmetro é: " . func_get_arg(2);
        }else{
            echo "Não existe este valor!<br/>";
        }
    }
    teste1('azul',3);
    teste1('azul',3,'casa',true);
?>
```

```
</p>  
</body>  
</html>
```

Ao observar o código, repare que, para obter o terceiro valor passado na função, é utilizado o índice 2, porque o primeiro valor é 0, tal como acontece com os arrays.

A visualização será:



Outra opção disponível com os parâmetros de uma função é forçar cada parâmetro a ser de um tipo de dado específico. Desta forma, se os parâmetros do tipo definido não forem passados para a função, o PHP irá lançar um erro, que no PHP5 é um erro fatal recuperável, ou seja, erro fatal porque não consegue ler os dados passados, mas recuperáveis porque percebe que foram passados dados, contudo, precisa da indicação de qual o tipo de dados que são para ler. Na versão 7 do PHP é do tipo `TypeError`.

Pode forçar um parâmetro a ser um destes tipos:

- **array**: o parâmetro deve ser um array.
- **bool**: o parâmetro deve ser do tipo boolean.
- **float**: o parâmetro deve ser um número com casas decimais.
- **int**: o parâmetro deve ser um número inteiro.
- **string**: o parâmetro deve ser uma string.
- **"Classe ou nome da interface"**: o parâmetro deve ser uma instância da classe ou interface especificada.
- **Self**: o parâmetro deve ser uma instância da mesma classe em que a função está definida.

Segue-se um exemplo de declaração de função que força os parâmetros a serem de um tipo específico:

```
<html>
<body>
<p>
<?php
function teste1( array $argumento1, int $argumento2){
echo "Eu forço o argumento1 a ser um array, e o valor é: ";
foreach ($argumento1 as $chave => $valor) {
echo $chave." => ".$valor.", ";
}
echo "Eu forço o argumento2 a ser um número inteiro, e o valor é:
".$argumento2;
}
teste1(array(0 => 'valor1', 1 => 'valor2' ),3);
?>
</p>
</body>
</html>
```

Ao executar o exemplo, a função mostrará:

← → ↻ localhost/exemploswdl/ud30ex6.php

Eu forço o argumento1 a ser um array, e o valor é: 0 => valor1, 1 => valor2, Eu forço o argumento2 a ser um número inteiro, e o valor é: 3

## 3. VALORES DEVOLVIDOS

---

A partir do PHP7, ao criar uma função é permitido determinar que tipo de valor será devolvido pela mesma. Os tipos de valores que podem ser especificados são os mesmos vistos anteriormente para determinar os tipos de parâmetros:

- **array:** a função irá devolver um array.
- **bool:** a função irá devolver um boolean.
- **float:** a função irá devolver um número com casas decimais.
- **int:** a função irá devolver um número inteiro.
- **string:** a função irá devolver uma string.
- **"Classe ou nome da interface":** a função irá devolver uma instância da classe ou interface especificada.
- **Self:** a função irá devolver uma instância da mesma classe em que está definida.

Segue-se um exemplo com a criação da função indicando o tipo de valor a ser devolvido:

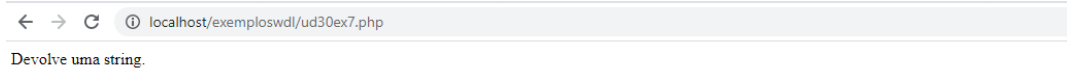
```
<html>
<body>
<p>
<?php
function teste1(): String{
```



```
        return "Devolve uma string.";
    }
    echo teste1();
?>
</p>
</body>
</html>
```

Neste exemplo, é indicado na função que deve ser devolvida uma string.

Ao executar o exemplo, a string devolvida pela função é exibida no ecrã da seguinte forma:



The screenshot shows a web browser window with the address bar displaying 'localhost/exemploswld/ud30ex7.php'. Below the address bar, the text 'Devolve uma string.' is displayed.

Devolve uma string.

## 4. FUNÇÕES VARIÁVEIS

---

O conceito de função variável é um conceito bastante complexo e o PHP é uma das poucas linguagens que o suporta.

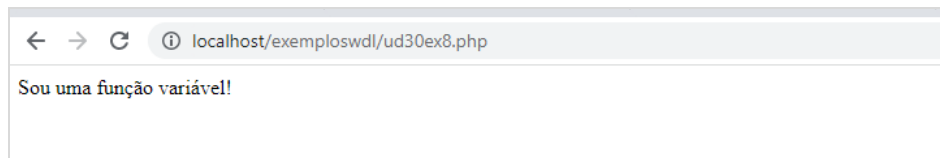
Uma função variável é uma variável criada que se tenta executar como se fosse uma função e, se o valor que esta variável contém existir como uma função, essa função será executada. No entanto, não funciona para funções PHP como `echo()`, `include()`, etc.

Segue-se um exemplo para clarificar:

```
<html>
<body>
<p>
<?php
    function teste(){
        echo 'Sou uma função variável!';
    }
    $var = 'teste';
    $var();
?>
</p>
</body>
```

```
</html>
```

Ao executar este exemplo, verá que a mensagem da função é exibida.



## 5. FUNÇÕES DO SISTEMA

---

A linguagem PHP possui funções predefinidas que auxiliam na programação do código. Têm sido utilizadas algumas delas ao longo desta unidade, como echo, return, etc.

A seguir, seguem-se alguns dos mais utilizados na programação:

### Funções matemáticas

---

- **abs**, para obter o valor absoluto de um número.

```
<?php  
echo abs(-10); // 10  
?>
```

- **ceil**, para arredondar as frações.

```
<?php  
echo ceil(10.9); // 11  
?>
```

- **round**, arredondando um número flutuante, pode tomar decimais como o segundo parâmetro.

```
<?php
echo round(1.45); // 1
echo round(1.55); //2
echo round(1.56,1); //1.6
?>
```

- **floor**, para arredondar uma fração para baixo.

```
<?php
echo floor(5.342); // 5
?>
```

- **max**, recebe uma array de números e devolve o maior.

```
<?php
echo max(10,20,3,1,45); // 45
?>
```

- **min**, como o anterior, mas devolve o número menor.

```
<?php
echo min(10,20,3,1,45); // 1
?>
```

- **mt\_rand**, cria um número aleatório, podendo passar pelos parâmetros o menor e o maior desejados na faixa.

```
<?php
echo mt_rand(10,20); //devolve um número entre 10 e 20
?>
```

- **sqrt**, devolve a raiz quadrada.

```
<?php
echo sqrt(9); //devolve 3
?>
```

- **intdiv**, executa uma divisão inteira dos números recebidos e devolve o resultado.

```
<?php
echo intdiv(5,2); //devolve 2
?>
```

---

## Funções de processamento de texto

---

- **chr**, devolve um carácter específico de acordo com o código ASCII passado pelo parâmetro.

```
<?php echo chr(65); //escreve a letra 'a' ?>
```

- **sprintf**, serve para imprimir uma string com um certo formato. Para definir o formato, deverá especificar dentro da string todos os parâmetros que pretende incluir com o símbolo '%' seguido pelo carácter que indica como o valor é exibido. Seguem-se os valores possíveis para imprimir uma string:
  - **b**: como um número binário.
  - **c**: como um carácter com valores ASCII.
  - **d**: como um número decimal com sinal.
  - **e**: como notação científica.
  - **E**: como notação científica maiúscula.
  - **u**: como um número decimal sem sinal.
  - **f**: como um float.
  - **o**: como um número octal.
  - **s**: como uma string.

- **x**: como um número hexadecimal minúsculo.
- **X**: como um número hexadecimal em maiúsculas.

```
<?php
sprintf('a cor favorita é %s', $cor);
?>
```

- **htmlentities**, converte todos os símbolos possíveis na sua entidade HTML, por exemplo, o “á” converte para “&acute;”. É muito útil exibir textos de acordo com a codificação da página web.

```
<?php
$frase = 'Aquele caminhão é<b>vermelho</b>';
echo htmlentities $frase;
//devolve: Aquele caminhão é b&gt;vermelho&lt;/b&gt;
?>
```

- **ltrim**, remove os espaços em branco do início da string.

```
<?php echo(ltrim(' olá')); //devolve 'olá' ?>
```

- **rtrim**, remove os espaços em branco do final da string.

```
<?php echo(rtrim(' olá ')); //devolve ' olá' ?>
```

- **trim**, remove os espaços em branco do início e do final da string.

```
<?php echo(trim(' olá ')); //devolve 'olá' ?>
```

- **md5**, calcula o hash md5 da string, ou seja, codifica a string com o algoritmo md5.

```
<?php echo md5('casa');
//devolve '202447d5d44ce12531f7207cb33b6bf7'
?>
```

- **ord**, devolve o número ASCII de um carácter.

```
<?php echo ord('A') //devolve 65 ?>
```

- **str\_replace**, com esta função são substituídos excertos de strings por outros.

- Pode ser utilizado com frases simples:

```
<?php echo str_replace('casa' , 'castelo' , 'vivo num casa');  
//devolve 'vivo num castelo'  
?>
```

- Ou para substituir grupos de palavras:

```
<?php  
$frase = 'vivo com os meus pais numa casa';  
$array = array('pais','numa','casa');  
$array2 = array('amigos','num','castelo');  
echo str_replace($array,$array2,$frase);  
//devolve 'vivo com os meus amigos num castelo'  
?>
```

- Ou para substituir strings e contar quantas vezes foram alteradas:

```
<?php  
$frase = str_replace('c','d','vivo num castelo', $ct);  
echo $frase; // devolve 'vivo num dastelo';  
echo $ct; //devolve 1  
?>
```

- **strlen**, devolve o comprimento da string.

```
<?php echo strlen('casa de madeira'); // devolve 15 ?>
```



- **stripos**, encontra a posição da última ocorrência da string independentemente do caso.

```
<?php echo stripos('a minha casa é na rua do carmo','ca');  
//devolve 25 ?>
```

- **strpos**, devolve a primeira posição da string pesquisada dentro de outra.

```
<?php php echo strpos('a minha casa é na rua do carmo','ca');  
//devolve 8 ?>
```

- **substr**, devolve uma parte da string, indicando por parâmetro de qual posição e quantos caracteres.

```
<?php echo substr('abcdef', 1, 3); //devolve 'bcd' ?>
```

- **ucfirst**, converte o primeiro carácter da string em maiúsculas.

```
<?php echo ucfirst('olá, amigo'); //devolve 'Olá, amigo' ?>
```

- **ucwords**, converte o primeiro carácter de cada palavra na string em maiúsculas.

```
<?php echo ucwords('olá, amigo'); //devolve 'Olá, Amigo' ?>
```

- **strtolower**, converte a string em minúsculas.

```
<?php echo strtolower('AmiGo'); //devolve 'amigo' ?>
```

- **strtoupper**, converte a string em maiúsculas.

```
<?php echo strtoupper('AmiGo'); //devolve 'AMIGO' ?>
```

- **print\_r**, exibe todo o conteúdo de um array no ecrã.

## Funções de calendário

---

- **cal\_days\_in\_month**, devolve o número de dias de um mês e ano passado no parâmetro, indicando o tipo de calendário no primeiro parâmetro passado.

```
<?php
echo cal_days_in_month(CAL_GREGORIAN, 8, 2003);
//devolve 31
?>
```

□ Os calendários possíveis são:

- CAL\_GREGORIAN (Gregoriano).
- CAL\_JULIAN (Juliano).
- CAL\_JEWISH (Judaico).
- CAL\_FRENCH (Republicano francês).

- **jddayofweek**, devolve o dia da semana em inglês, passando uma data no formato juliano e o modo.

```
<?php echo jddayofweek ("12/25/2010" , 1 ); // devolve 'Saturday' ?>
```

0	Gregoriano abreviado	Jan, Feb...
1	Gregoriano	January, February...
2	Juliano abreviado	Jan, Feb...
3	Juliano	January, February...
4	Judaico	Tishri, Heshvan
5	Republicano francês	Vendemiaire, Brumaire...

- **jdmonthname**, devolve o nome de um mês, passando uma data e o modo, tal como o anterior.

```
<?php echo jdmonthname ("12/25/2010" , 1 ); //devolve 'December' ?>
```

## Funções de data/hora

- **checkdate**, são passados mês, dia e ano pela função e esta devolve true, se for uma data válida.

```
<?php echo checkdate(2,30,2010); //devolve 'false'??>
```

- **date**, formata a data e hora locais; é utilizado para obter a data e hora atuais com o formato desejado ou passando uma data específica como parâmetro para atingir o formato desejado.

```
<?php
// alguns formatos de data
echo date("F j, Y, g:i a").'<br/>';
echo date("m.d.y").'<br/>';
echo date("j, n, Y").'<br/>';
echo date("Ymd").'<br/>';
echo date('h-i-s, j-m-y').'<br/>';
echo date('jS').'<br/>';
echo date("D M j G:i:s T Y").'<br/>';
echo date("H:i:s").'<br/>';
?>
```

□ Resultado:

```
March 7, 2021, 3:51 pm
03.07.21
7, 3, 2021
20210307
03-51-04, 7-03-21
7th
SunMar7 15:51:04UTC 2021
15:51:04
13:37:26
```

- **time**, devolve o momento atual calculado em segundos desde 1 de janeiro de 1970.

```
<?php
$seguinte = time() + (7 *24 *60 *60);
echo 'Hoje ' .date('Y-m-d');
echo 'Proxima semana ' .date('Y.m.d',$seguinte);
?>
```

- **strtotime**, esta função permite converter uma descrição da string (em inglês) para uma data em segundos a partir de 1 de janeiro de 1970.

```
<?php
echo date('Y-m-d',strtotime("now")). "<br>";
echo date('Y-m-d',strtotime("10 September 2000")). "<br>";
echo date('Y-m-d',strtotime("+3 day")). "<br>";
echo date('Y-m-d',strtotime("+2 week")). "<br>";
echo date('Y-m-d',strtotime("+1 week 2 days 4 hours 2 seconds")).
"<br>";
echo date('Y-m-d',strtotime("next Thursday")). "<br>";
echo date('Y-m-d',strtotime("last Monday")). "<br>";
?>
```

□ Devolve:

```
2021-03-07
2000-09-10
2021-03-10
2021-03-21
2021-03-16
2021-03-11
2021-03-01
```

## CONCLUSÃO

---

As funções permitem criar excertos de código independentes e, consequentemente, organizar o código da página web e trabalhar de forma não linear.

Graças às funções é possível criar páginas web cada vez mais complexas e com maior interação com o utilizador.



## AUTOAVALIAÇÃO

---

1. Qual é a palavra reservada para a criação de funções em PHP?
  - a) function.
  - b) função.
  - c) class.
  - d) classFunction.
  
2. O que é utilizado para devolver os parâmetros de uma função?
  - a) return.
  - b) send.
  - c) strret.
  - d) returnStr.
  
3. A linguagem PHP diferencia maiúsculas de minúsculas em nomes de funções?
  - a) Não.
  - b) Sim.
  - c) Apenas em funções recursivas.
  - d) Apenas em funções dentro das classes.

- 4. O que significa passar um parâmetro por valor?**
- a) Passar o identificador para a variável e modificar o seu valor.
  - b) Passar os valores como números ou texto, mas não como variáveis.
  - c) Passar o valor para a função se é uma variável ou um valor escrito, mas sem modificar o seu valor se for uma variável.
  - d) Passar o valor para a função se é uma variável ou um valor escrito, modificando o seu valor se for uma variável.
- 5. Qual é o símbolo utilizado para passar uma variável como parâmetro por referência?**
- a) Exclamação (!).
  - b) Percentagem (%).
  - c) Cardinal (#).
  - d) 'E' comercial (&).
- 6. O que é que está a ser indicado ao passar um parâmetro com código '\$num = 5'?**
- a) Que o valor da variável será sempre 5.
  - b) Que no final da função o valor da variável é igual a 5.
  - c) Que, se não passarmos um parâmetro para a função, a variável valerá 5.
  - d) Não é possível passar uma variável como parâmetro.
- 7. Qual é a função PHP que permite saber o número de parâmetros recebidos na função?**
- a) num\_args.
  - b) func\_num\_args.
  - c) get\_num\_args.
  - d) count\_num\_args.



8. O que é utilizado para selecionar um parâmetro recebido na função sem definição prévia de parâmetros?
- a) `func_get_arg(n)`.
  - b) `get_arg(n)`.
  - c) `$_GET[n]`.
  - d) `return`.
9. O que é uma função variável?
- a) Uma função que pode alterar parâmetros.
  - b) Uma função que altera os valores das variáveis recebidas.
  - c) Uma variável que contém o nome de uma função que é executada quando essa variável é chamada como uma função.
  - d) Uma função com o id variável.
10. Qual é a função que permite arredondar as frações?
- a) `ceil`.
  - b) `round`.
  - c) `floor`.
  - d) `abs`.



## SOLUÇÕES

---

1.	a	2.	a	3.	b	4.	c	5.	d
6.	c	7.	b	8.	a	9.	c	10.	a

## PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

---

Consulte a seguinte página web para consultar todas as funções do PHP:

- [http://www.php.net/manual/pt\\_BR/funcref.php](http://www.php.net/manual/pt_BR/funcref.php)

Consulte a seguinte página web para consultar todos os formatos de data possíveis em PHP:

- [http://www.php.net/manual/pt\\_BR/datetime.formats.php](http://www.php.net/manual/pt_BR/datetime.formats.php)

## BIBLIOGRAFIA

---

- Cabezas, L. M. (2010), *Php 5*. Madrid: Anaya Multimedia.
- The PHP Group (2001), “Funções”. Disponível em:  
[http://www.php.net/manual/pt\\_BR/language.functions.php](http://www.php.net/manual/pt_BR/language.functions.php)  
Consultado a 9 de março de 2021.

