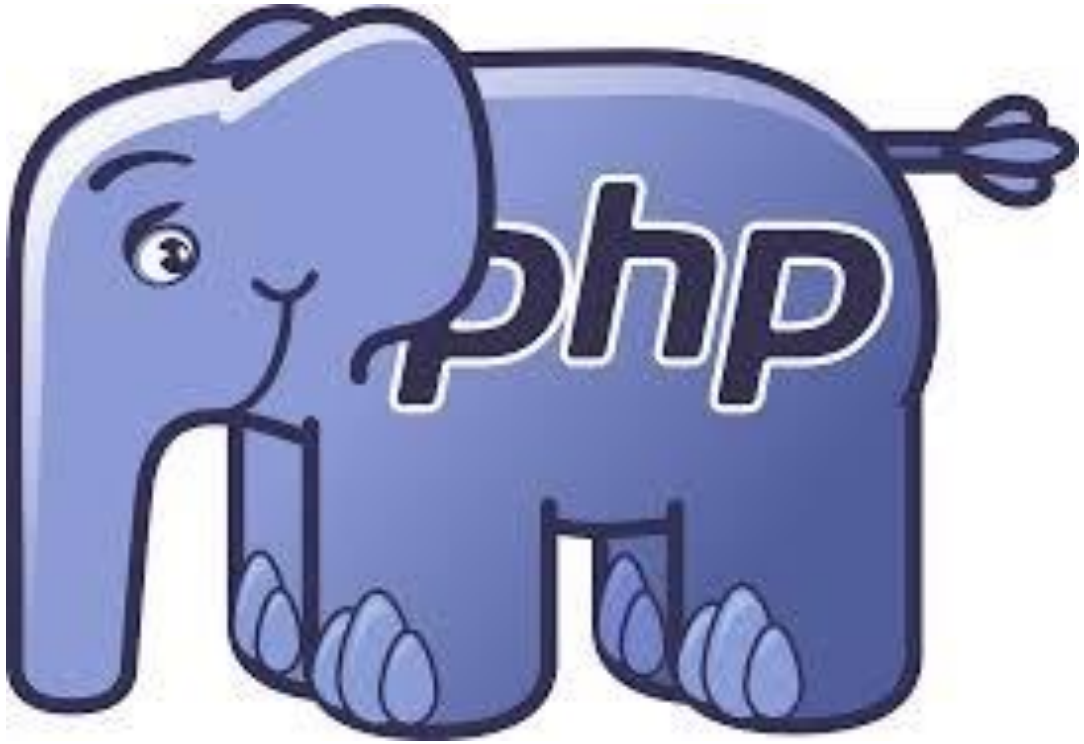

AULA 4

EXPLICADORES.NET

@explicadoresnet

foreach

O comando *foreach* funciona apenas na versão 4 do PHP. O PHP3 não oferece esse suporte. O *foreach* nos oferece uma maneira mais fácil de "navegar" entre os elementos de um array. Observe as duas sintaxes possíveis:

```
foreach ($nome_array as $elemento)
{
    comandos
}
```

ou

```
foreach ($nome_array as $chave => $valor)
{
    comandos
}
```


```php
<?php
$vetor = array (1, 2, 3, 4);
foreach ($vetor as $v)
{
    print "O valor atual do vetor é $v. <br>";
}
$a = array ( "um" => 1, "dois" => 2, "tres" => 3 );
foreach($a as $chave => $valor) {
    print "\$a[$chave] => $valor.<br>";
}
?>
```

O programa apresentado mostrará na tela todos os valores do array $*vetor*, e depois mostrará todos as chaves e valores do array $*a*. O segundo *foreach* desse exemplo mostrará o seguinte:

```
$a [um] => 1.
$a [dois] => 2.
$a [tres] => 3.
```

$vetor = array (1,2,3,4)
foreach ($vetor as $chave => $valor)
echo $chave;
if ($valor==3){ break }

function executa ($param){
$resultado = 0;
foreach ($param as $valor){
$resultado+=valor;
}
return $resultado;
}
echo executa (array(3,5,10,2))

Funções para manipulação de arquivos

Essas funções são responsáveis pelas diversas operações que podemos realizar sobre os arquivos. O PHP nos oferece dezenas de funções capazes de trabalhar com o filesystem (sistema de arquivos) do sistema operacional. As principais operações que podemos realizar sobre um arquivo são: abertura, leitura, escrita e fechamento. Portanto, veremos com maiores detalhes as funções que realizam essas tarefas.

fopen

Para abrir um arquivo, utilizamos a função *fopen*, que possui a seguinte sintaxe:

```
recurso fopen (string nome_arquivo, string modo [, int
   usar_include_path [, recurso contexto]])
```

Parâmetro	Descrição
nome_arquivo	Nome do arquivo a ser aberto, que pode ser local ou remoto. Se for fornecido um URL, a abertura só irá funcionar se a diretiva *allow_url_fopen* estiver habilitada no *php.ini*.
modo	Modo de acesso ao arquivo. A lista dos modos disponíveis será apresentada logo a seguir.
usar_include_path	Indica se o arquivo deve ser procurado nos diretórios especificados na diretiva *include_path* do *php.ini*.
contexto	Permite a definição de um contexto, que consiste em um conjunto de parâmetros que modificam o comportamento do arquivo. O suporte a contextos foi adicionado no PHP 5.

O segundo parâmetro da função *fopen* é o *modo*, que pode possuir os seguintes valores:

Modo	Descrição
'r'	Abre somente para leitura, posicionando o ponteiro no início do arquivo.
'r+'	Abre para leitura e escrita, posicionando o ponteiro no início do arquivo.
'w'	Abre somente para escrita, posicionando o ponteiro no início do arquivo e deixando-o com tamanho zero. Se o arquivo não existir, tenta criá-lo.
'w+'	Abre para leitura e escrita, posicionando o ponteiro no início do arquivo e deixando-o com tamanho zero. Se o arquivo não existir, tenta criá-lo.
'a'	Abre somente para escrita, posicionando o ponteiro no final do arquivo. Se o arquivo não existir, tenta criá-lo.
'a+'	Abre para leitura e escrita, posicionando o ponteiro no final do arquivo. Se o arquivo não existir, tenta criá-lo.
'x'	Cria e abre um arquivo somente para escrita, posicionando o ponteiro no início do arquivo. Se o arquivo já existir, retorna falso (*FALSE*) e gera um erro do tipo *E_WARNING*. Disponível desde a versão 4.3.2 do PHP, esse modo é usado apenas em arquivos locais.
'x+'	Cria e abre um arquivo para leitura e escrita, posicionando o ponteiro no início do arquivo. Se o arquivo já existir, retorna falso (*FALSE*) e gera um erro do tipo *E_WARNING*. Disponível desde a versão 4.3.2 do PHP, esse modo é usado apenas em arquivos locais.

Existe ainda a possibilidade de usar as letras 'b' ou 't' como último caractere do parâmetro *modo*. A letra 'b' força o uso do modo binário, devendo ser utilizada em sistemas que diferenciam arquivos nos formatos binário e texto. A letra 't' é usada em sistemas Windows para traduzir os caracteres de quebra de linha (\n) para \r\n. Veja alguns exemplos de uso da função *fopen*:

```
<?php
$ponteiro = fopen ("/home/juliano/teste.txt", "r");
$ponteiro = fopen ("/home/juliano/teste2.txt", "wb");
$ponteiro = fopen ("/home/juliano/teste2.txt", "a+");
$ponteiro = fopen ("http://www.teste.com.br", "r");
?>
```


fclose

Para fechar um arquivo, utilizamos a função *fclose*, que possui a seguinte sintaxe:

```
bool fclose (recurso ponteiro_arquivo)
```

Essa função retorna *true* se o arquivo foi fechado com sucesso e retorna *false* se houver alguma falha. O parâmetro passado para a função *fclose* deve conter a variável para a qual foi atribuído o resultado da função *fopen*, ou seja, o ponteiro (handle) para o arquivo que foi aberto. Exemplo:

```
<?php
$ponteiro = fopen('arquivo.txt', 'r');
...
fclose($ponteiro);
?>
```

fread

Uma das formas de ler dados de um arquivo é utilizar a função *fread*, que permite especificar a quantidade de informações a serem lidas. Sua sintaxe é a seguinte:

```
string fread (recurso ponteiro_arquivo, int tamanho)
```

Essa função lê o número de bytes especificado no parâmetro *tamanho* a partir da posição atual do ponteiro definido pelo primeiro parâmetro. A leitura termina quando o número de bytes especificado é lido ou o fim do arquivo (*EOF – End Of File*) é alcançado. Por exemplo:

fgets

Também realiza a leitura de um arquivo, mas lê no máximo uma linha. É bastante utilizado quando queremos trabalhar individualmente com cada linha do arquivo. Sua sintaxe é a seguinte:

```
string fgets (recurso ponteiro_arquivo [, int tamanho])
```

A leitura é feita até que seja lido o número de bytes especificados em *tamanho*, ou quando terminar a linha atual do arquivo (caracter \n), ou quando o arquivo chegar ao seu final. Se não for especificado o parâmetro *tamanho*, será utilizado o valor padrão, que é 1024 bytes (1k). Como exemplo, considere o programa *linha.php* apresentado a seguir.

linha.php

```
<?php
$ponteiro = fopen ("teste.txt", "r");
$linha = fgets($ponteiro, 4096);
echo $linha;
fclose ($ponteiro);
?>
```

Esse programa lê a primeira linha de um arquivo, exibindo-a na tela. Note que foi utilizado um valor bem alto (4096) para garantir a leitura da linha inteira. Se a função *fgets* fosse chamada novamente, seria retornada a segunda linha do arquivo, e assim por diante.

fwrite

Para escrever dados em um arquivo com o PHP, utilizamos a função *fwrite*, que possui a seguinte sintaxe:

```
int fwrite (recurso ponteiro_arquivo, string string [, int tamanho])
```

Esta função escreve o conteúdo do parâmetro *string* no arquivo referenciado pelo parâmetro *ponteiro_arquivo*. O parâmetro *tamanho* é opcional. Se for informado, a escrita irá parar após serem atingidos o número de bytes especificado. Veja um exemplo de utilização dessa função:

escrita.php

```
<?php
$conteudo = "Este texto será escrito no arquivo!";
$ponteiro = fopen ("arquivo.txt", "w");
fwrite($ponteiro, $conteudo);
fclose ($ponteiro);
?>
```


fopen - Abre um arquivo.

```
$arq = fopen(“/docs/texto.txt”, "r"); // apenas para leitura
```

fclose - Fecha um arquivo aberto.

```
fclose($arq);
```

fread - Lê uma quantidade de bytes específica.

```
$txt = fread($arq, 30); // lê 30 bytes
```

fgets - Lê bytes até chegar ao fim da linha.

```
$txt = fgets($arq, 5000);
```

fgetss - Lê byte, eliminando tags HTML e PHP.

```
$txt = fgetss($arq, 5000);
```

fwrite - Grava em arquivos abertos com permissão.

```
fwrite($arq, “Salvando novos dados”);
```


Outras funções de Arquivos

basename - Retorna o nome do arquivo.

chmod - Modifica as permissões do arquivo.

chown - Modifica o proprietário do arquivo.

copy - Copia um arquivo.

delete - Apaga um arquivo.

ftruncate - Apaga o conteúdo, mantendo o arquivo.

feof - Verifica se o fim do arquivo já foi atingido.

file_exists - Verifica se o arquivo existe.

filesize - Indica o tamanho de um arquivo em bytes.

```php
<?php
$arquivo = "contador.txt";  // arquivo do contador
if(file_exists($arquivo))   // se existe, lê o valor atual e o incrementa
{
```


PHP e formulários HTML

Talvez esta seja uma das principais razões que levou você a adquirir este livro: como criar um formulário para os usuários de meu site preencherem? Como faço para enviar essas informações para o meu programa PHP? E como devo tratá-las dentro do meu programa? Chegou a hora de conhecer as respostas para todas essas perguntas. Após a leitura deste tópico, você estará pronto para criar formulários que deixarão seu site muito mais interativo.

Para tornar esse formulário útil, devemos informar ao browser para onde devem ser enviadas as informações. Você utilizará os formulários para enviar dados aos seus programas PHP, então o correto é informar ao browser qual programa receberá esses dados. Isso é feito com a opção *action*, utilizada na tag *form* do HTML. Veja o exemplo a seguir:

```
<form action="recebe_dados.php">
    <p>Digite seu e-mail: <input type="text" name="email" size="20"></p>
    <p><input type="submit" value="Enviar!" name="enviar"></p>
</form>
```


Opção	Descrição
name	Informa qual é o nome do campo.
value	Informa um valor padrão para o campo.
size	Informa o tamanho do campo exibido na tela.
maxlength	Informa o número máximo de caracteres que pode ser digitado no campo.
type	Informa qual é o tipo do campo de entrada de dados.

Os valores possíveis para a opção *type* da tag *input* são mostrados na tabela a seguir:

Valor	Descrição
text	Mostra uma caixa de texto de uma linha, e permite a entrada de valores numéricos ou alfanuméricos.
password	Utilizado para a digitação de senhas. São mostrados asteriscos (*) no lugar dos caracteres digitados, mas a informação é enviada normalmente.
hidden	É um campo escondido. Não aparece na tela. Podemos utilizá-lo para passar informações aos programas que recebem os dados. Veremos mais adiante que esses campos são de grande utilidade.

Valor	Descrição (continuação)
select	Mostra uma lista de seleção (também conhecida como drop-down).
checkbox	Exibe uma caixa de seleção, que pode ser marcada ou desmarcada.
radio	São botões de seleção, em que o usuário escolhe uma entre várias opções disponíveis.
textarea	Caixa de texto com várias linhas.
file	Permite o envio de arquivos.
submit	Botão que aciona o envio dos dados do formulário.
image	Tem a mesma função que o *submit*, mas utiliza uma imagem em vez do botão tradicional do formulário.
reset	Limpa todos os campos de um formulário e retorna ao valor-padrão (se existir).

Formulário

Nome:

E-mail:

Telefone:

Idade: 18

Sexo:

M **F**

Tipo de navagador utilizado:

Google Chrome **Internet Explorer** **Firefox** **Safari** **Outros**

Mensagem:

Digite aqui sua mensagem....

Enviar Limpar

1 - radio ( ) Usado para acionar o envio de dados do formulário.
2 - text ( ) Permite a entrada de dados numéricos ou alfa-numéricos.
3 - select ( ) Caixa de seleção que pode ser marcada ou desmarcada.
4 - submit ( ) Mostra uma lista de seleção do tipo drop-down.
5 - checkbox ( ) Permite que o usuário selecione apenas um elemento

a) 1–2–3–5–4
b) 4–2–3–5–1
c) 1–3–5–2–4
d) 4–3–5–2–1
e) 4 – 2 - 5- 3 - 1

Enviando as informações para um programa PHP

Vimos no tópico anterior que para especificar qual programa PHP receberá os dados do formulário utilizamos a opção *action* da tag *form* do HTML. Exemplo:

```
<form action="processa_dados.php">
```

Resta saber como esses dados são passados ao programa PHP. Existem dois métodos de passagem de parâmetros: *GET* e *POST*. No caso de um formulário, o tipo de método a ser utilizado é especificado na opção *method* da tag *form*. Exemplo:

```
<form action="processa_dados.php" method="POST">
```

Vamos ver com detalhes cada um deles para que você entenda a diferença.

Método GET

Esse é o método-padrão para o envio de dados. Se no momento da criação de um formulário nenhum método for especificado na opção *method* da tag *form*, estaremos utilizando o método *GET* para o envio dos dados.

Nesse método, os dados serão enviados juntamente com o nome da página (na URL) que processará os dados recebidos. Considere o seguinte formulário como exemplo:

```
<form action="recebe_dados.php">
    <p>Digite seu nome: <input type="text" name="nome" size="30"></p>
    <p>Digite sua idade: <input type="text" name="idade" size="3"></p>
    <p><input type="submit" value="Enviar!" name="enviar"></p>
</form>
```

Note que esse formulário não mostra a opção *method*, portanto o padrão é adotado (method="GET"). Suponha que preenchemos o campo *nome* com o valor *Joaquim*, e o campo idade com o valor *20*. Logo após clicarmos no botão "Enviar!", o endereço ativado será o seguinte:

```
http://www.seusite.com.br/recebe_dados.php?nome=Joaquim&idade=20
```

Os campos do formulário serão passados como parâmetros após o endereço de destino. O caractere ? representa o início de uma cadeia de variáveis, e o símbolo & identifica o início de uma nova variável. As variáveis e seus respectivos valores são separadas pelo caractere =.

Existem alguns inconvenientes de utilizar o método *GET*. Primeiramente porque há um limite de caracteres que podem ser enviados (em torno de 2.000 caracteres). Outro problema é que o usuário enxergará todos os parâmetros por meio da barra de endereço do browser, o que não é muito agradável. Para resolver esses problemas existe o método *POST*.

Em compensação, o método *GET* possui uma grande vantagem: além de enviar informações via formulários, esse método pode ser utilizado também para a passagem de parâmetros por meio de links. Imagine, por exemplo, uma loja virtual, onde há um link para cada produto. Certamente cada link passa como parâmetro um número identificador do produto, para ser tratado por determinado programa. O link para um produto específico poderia ser o seguinte:

```
http://www.lojinhadojoao.com.br/produto.php?id_produto=50
```


```
http://www.lojinhadojoao.com.br/produto.php?id_produto=50
```

Dessa forma, um programa chamado *produto.php* receberia o número identificador do produto e realizaria uma consulta ao banco de dados para buscar informações detalhadas, como descrição, preço etc. Podemos, se necessário, passar mais de um parâmetro através do método *GET*. Para isso basta utilizar o símbolo & fazendo a separação. Por exemplo, o link para consulta de uma subcategoria da loja poderia ser o seguinte:

```
http://www.lojinhadojoao.com.br/produto.php?categoria=2&subcategoria=5
```

Somente com o método *GET* podemos passar parâmetros por links. O método *POST* trabalha somente com formulários.

Método POST

Para utilizar o método *POST* devemos utilizar opção *method* da tag *form* para informar ao browser. Exemplo:

```
<form action="recebe_dados.php" method="POST">
    <p>Digite seu nome: <input type="text" name="nome" size="30"></p>
    <p>Digite sua idade: <input type="text" name="idade" size="3"></p>
    <p><input type="submit" value="Enviar!" name="enviar"></p>
</form>
```

Ao contrário do método *GET*, que envia os dados por uma cadeia de variáveis após o endereço-destino, o método *POST* envia os dados do formulário por meio do corpo da mensagem encaminhada ao servidor.

Como os dados são enviados no corpo da mensagem, quando o usuário clicar no botão "Enviar!" ele não verá em sua barra de endereços aquele endereço enorme contendo uma cadeia de variáveis. Ele verá apenas o endereço do programa ativado:

```
http://www.seusite.com.br/recebe_dados.php
```


Utilizar os arrays superglobais predefinidos pelo PHP. Existem dois arrays que o PHP utiliza: um para armazenar os valores enviados pelo método *GET* e outro para armazenar informações enviadas pelo método *POST*. Esses arrays são o $_*GET* e o $_*POST*.

Nesse caso, o nome dos campos do formulário é usado como chave associativa, e o valor dos campos é armazenado como os valores do array. Se usássemos, por exemplo, o método *POST* para enviar um formulário com os campos *nome* e *email*, dentro do programa PHP acessaríamos esses dados da seguinte maneira:

```
$_POST["nome"]
$_POST["email"]
```

Se o método utilizado fosse o *GET* usaríamos:

```
$_GET["nome"]
$_GET["email"]
```


Cookies e sessões

Se você precisa manter informações sobre seus usuários enquanto eles navegam pelo seu site, ou até quando eles saem do mesmo, é importante que você saiba lidar com cookies ou com sessões. Cookies são arquivos-texto que podem ser armazenados no computador do usuário, para serem recuperados posteriormente pelo servidor no qual seu site está hospedado. Sessões são recursos que podemos utilizar para manter uma conexão com o usuário, enquanto ele estiver navegando no site. Veremos com detalhes esses dois mecanismos nos próximos tópicos.

Utilizando cookies

Cookie é um arquivo-texto que podemos armazenar no computador do usuário, para ser recuperado posteriormente pelo servidor. Um cookie é formado por um par nome/valor, ou seja, possui um nome pelo qual ele é referenciado e um valor associado a esse nome. Podem ser utilizados em qualquer aplicação que necessite compartilhar dados entre diferentes páginas, ou até entre diferentes acessos (em dias e horários distintos).

Os cookies podem ser mantidos na máquina do usuário por vários dias, ao contrário das sessões, que mantêm os dados somente enquanto o usuário permanecer no seu site.

Se for utilizado somente o parâmetro *nome*, o servidor tentará excluir o cookie do computador do usuário. Portanto, para definir um cookie devemos utilizar no mínimo os parâmetros *nome* e *valor*. Exemplo:

```
setcookie ("nome","Juliano");
```

Para excluir o cookie criado anteriormente basta executar o comando:

```
setcookie ("nome");
```

Para criar, por exemplo, um cookie válido por 2 dias (48 horas) podemos utilizar como auxílio a função *time* do PHP. Exemplo:

```
setcookie ("nome", "Juliano", time()+172800);
```


Criando uma sessão no PHP

No PHP uma sessão pode ser criada de forma manual ou automática. A criação manual pode ser feita de forma explícita (função *session_start*) ou de forma implícita (ao registrar uma variável com a função *session_register*). A forma automática consiste em habilitar a diretiva *session.auto_start* do arquivo *php.ini*. Assim, sempre que um usuário entrar em seu site será automaticamente criada uma sessão.

A função *session_start* serve tanto para criar uma sessão como para restaurar os dados uma sessão com base no identificador corrente (passado pelo método *GET*, *POST* ou por cookie). Essa função não possui parâmetros.

```
bool session_start (vazio)
```

Se você estiver usando o método dos cookies para armazenar o identificador da sessão, deve chamar a função *session_start* antes de qualquer saída produzida pelo browser.

```
<?php
session_start();
...
?>
```


– Utilizando-se linguagem PHP, qual a instrução correta para destruir uma variável Y, ou seja, liberar a memória ocupada por ela, fazendo com que ela deixe de existir?

a) gettype($Y)
b) empty($Y)
c) echo($Y)
d) unset($Y)

- Qual função é caracterizada como a mais simples para ordenação de arrays no PHP?

a) sort(array);
b) rsort(array);
c) asort(array);
d) ksort(array);

– Qual função de PHP retorna o código ASCII correspondente ao caractere fornecido?

a) print(string);
b) ord(string);
c) chr(código ascii);
d) echo(string1, string[argn]...);

Abª

– Quais são os parâmetros utilizados pela função *mail* na linguagem PHP?

a) remetente, headers, texto, destinatário
b) destinatário, assunto, mensagem, headers
c) destinatário, remetente, mensagem, headers
d) remetente, mensagem, assunto, destinatário

– O que faz a função *gettype* no PHP?

a) Verifica se uma variável possui valor.
b) Retorna o tipo da variável.
c) Testa o tipo da variável.
d) Destrói uma variável.

– A respeito da função *fopen* na linguagem PHP, é correto afirmar que esta função

a) pode ser usada para ler o conteúdo de um arquivo binário.
b) retorna um *string* com o conteúdo do arquivo.
c) retorna *false* em caso de erro e um identificador do arquivo em caso de sucesso.
d) faz uma cópia de arquivo, desde que o usuário possua as permissões necessárias.

– No que diz respeito à manipulação de números na linguagem PHP, qual expressão gera um número aleatório?

a) “rand”
b) “round”
c) “sqrt”
d) “min”

– Considerando que o PHP está sendo utilizado para manipulação de arquivos, relacione a coluna da direita com a da esquerda e depois; marque a alternativa que apresenta a sequência correta. Alguns números poderão ser utilizados mais de uma vez e outros poderão não ser usados.

1- *fread(arquivo, tamanho);*	( ) Esta função grava em um arquivo o conteúdo do segundo parâmetro.
2- *fwrite(arquivo, conteudo, tamanho);*	( ) Esta função retorna um inteiro com o tamanho do arquivo, em *bytes*, ou *false* em caso de erro.
3- *fopen*	( ) Utiliza um *cache*.
4- *filesize(arquivo);*	( ) Função que pode ser usada para os modos de leitura e/ou escrita.
5- *mail(destinatario, assunto, mensagem, headers);*	( ) Função que retorna um *string* com o conteúdo do arquivo.

a) 5 - 3 - 4 - 4 - 1
b) 2 - 4 - 3 - 3 - 1
c) 2 - 4 - 4 - 3 - 1
d) 1 - 4 - 4 - 3 - 5

– Considere a função que se segue, utilizada na linguagem PHP, e assinale a alternativa que corresponde à ação praticada por esta função.

array range(minimo, máximo);

a) "Embaralha" o *array*, ou seja, troca as posições dos elementos aleatoriamente e não retorna valor algum.
b) Cria um *array* a partir dos parâmetros fornecidos. É possível fornecer o índice de cada elemento. Esse índice pode ser um valor de qualquer tipo, e não apenas de inteiro.
c) Cria um *array* cujos elementos são os inteiros pertencentes ao intervalo fornecido, inclusive. Se o valor do primeiro parâmetro for maior do que o do segundo, a função retorna *false* (valor vazio).
d) Retorna um valor inteiro contendo o número de elementos de um *array*.

– Considerando o uso do PHP para manipulação de números, assinale a assertiva verdadeira.

a) *"round"* é a função que gera um número aleatório.
b) *"rand"* é a função que arredonda um número.
c) *"sqrt"* é a função que retorna a raiz quadrada.
d) *"floor"* é a função que arredonda frações para cima.