CURSO PHP ALURA – INTRODUÇÃO AO PHP

INSTALAR O PHP:

Baixar a versão mais nova, descompactar o arquivo em uma pasta no C:, ir até as variáveis de ambiente e adicionar a pasta no **PATH.**

PHP.INI → Contém diversas configurações do PHP, como quais extensões vem habilitadas, como se conectar com o banco de dados específico, manipular algum tipo de dado, etc.

Escolher entre **development** ou **production** e renomear para php.ini.

ALGUNS COMANDOS:

PHP -A → Abre terminal interativo.

No final de cada comando, colocar **; (ponto e virgula).**

VAR_DUMP() → Mostra array com mais detalhes
PRINT_R() → Mostra array com menos detalhes
COUNT($VAR) → Quantidade de itens

QUIT → Sair do PHP

// ou **/* */ →** Adicionar comentários

$var → Declarar uma variável

. (ponto) → Concatenação
Exemplo:
echo $idade **.** "\n";

GETTYPE → Fala o tipo de variável que foi passado por parâmetro.

' ' → Aspas simples não lê variáveis, necessita concatenar. **Aspas simples lê tudo como strings.**

" " → Aspas duplas não necessita de concatenação.

"\n" → Quebra linha | **.PHP_EOL == "\n"**

"\t" → Parágrafo

|| ou **OR →** OU
&& ou **AND →** AND

Exemplo:

```
$var = 1;

while ($var <10) {
$var++;
}
```

Ou:

```
for ($i = 0 ; $i < 10 ; i++) {
condição
}
```

continue; → Continua o código dependendo da aplicação

break; → Para o código dependendo da aplicação

Exemploasd

```
if ($i < 0) {
condição
}
```

OPERADORES DE COMPARAÇÃO:

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

OPERADORES LÓGICOS:

&& ou **AND** → AND

|| ou **OR** → OU

XOR → Exclusivo

0 0 = 0
0 1 = 1
1 0 = 1
1 1 = 0

FOREACH ($array AS $valor) → Para cada item do array, ele atribui na variável $valor

Exemplo:

FOREACH ($array AS $indice => $valor)

Ou:

FOREACH ($array [“indice”] AS $subindice => $valor);

FUNCTION:

```
function calcularMedia ($nome, $n1, $n2, $n3, $n4) {

    $media = ($n1, $n2, $n3, $n4) / 4;

    if ($media < 5) {

        echo “$nome foi reprovado com média $media”;
    } else {

        echo “$nome” foi aprovado com a média $media;
    }
}

calcularMedia (“Elbert”, 4, 7, 9, 10);
```

CURSO PHP ALURA – AVANÇADO COM PHP

ARRAY → $ var [] → Sempre separar os itens por **,** (virgula).

Exemplo:

```
$var = [
        “titular” => “Elbert”,
        “saldo” => 1000
];
```

Ou:

```
$var = [
        123456789 => [
                “titular” => “Elbert”,
                “saldo” => 1000
        ];
        123456543 => [
                “titular” => “João”,
                “saldo” => 0
        ]
];
```

FOREACH:

Exemplo:

```
foreach ($contas AS $indice => $valor){
        echo “$indice = $valor[“titular”]”
};
```

$var[] = 20 → Com o índice não preenchido, o PHP entende que o valor da array vai para o próximo nº de índice automaticamente.

FUNCTION:

Exemplo:

```
function exibeMensagem ($mensagem) {
        echo ($mensagem) . “ \n”;
}

exibeMensagem (“Escrever a mensagem aqui”);
```

DEVOLVENDO PARÂMETROS:

Retorna o valor obtido no parâmetro e salva na var **$resultado.**

Exemplo:

```
function adiciona ($x) {
        return $x + 2;
        };

$resultado = adiciona(10);
echo $resultado              // $resultado vai retornar 12.
```

FUNÇÃO SUB-ROTINA → Executa e **não** devolve valor, já uma **função devolve.**

Dentro de uma string com aspas duplas, se coloca o indice de uma array sem aspas, apenas com conchas

Exemplo:

"$indice[valor]";

Também pode se utilizar com **chaves**

Exemplo:

{$indice ['valor']};** e não **{$indice[valor]}

REQUIRE E INCLUDE:

Caso o arquivo não for encontrado, **require** pausa a execução do programa, enquanto o **include** permite que continue.

LIST:

Exemplo:

LIST ("**titular**" => $**titular**, "**saldo**" => $**saldo**) **=** $valor
titular = índice
$titular = valor atribuido na array
$saldo = valor.

Ou:

["titular" => $titular, saldo => "$saldo"] = $valor

→ Pegar o exemplo de **foreach** para agregar **List**

UNSET → Remove uma variável da memória.

INICIAR UM SERVIDOR LOCAL:

No CMD, utilizar **php -S localhost:porta** (exemplo: 8000/ 8080), este comando inicia um servidor local no nosso ambiente de trabalho.

Utilizar esse comando na pasta do arquivo desejado.

CLIENTE → REQUISIÇÃO → SERVIDOR → RESPOSTA → CLIENTE

PHP NO HTML:

Abrir **<?php** para indicar no código html que vai ler um arquivo php. Fechar a tag com **?>**

<?= " Texto aqui **" ?>** → Serve como um echo.

LER ARQUIVO PHP:

Para ler um arquivo php no powershell, ir até o diretório e **php nomearquivo.php**

CURSO PHP ALURA – ORIENTAÇÃO A OBJETOS
CLASSES, METODOS E ATRIBUTOS

SRC → Source (código fonte)

REQUIRE (nomeArquivo) → Para abrir no terminal

Exemplo:
```
require 'src/conta.php';
```

CLASS:

```
Class Conta {                          // FORMA DO BOLO
    //definir dados                    // PLANTA DA CASA
}
```

$umaNovaConta = new Conta ();

$umaNovaConta → Lugar onde o bolo foi guardado (endereço)
$ Conta () → Bolo pronto.

INSTÂNCIA ← SINÔNIMO → OBJETO

public $CPF → Atributo

```
class Conta {
    public $cpf;
    public $nome;
    public $saldo;
}

$primeiraConta = new Conta ();
$primeiraConta → cpf = 1234567890;
$primeiraConta → nome = "Elbert";
$primeiraConta → valor = 0;
```

Class é a forma do bolo.
Objeto é o bolo em si.
Quando criamos uma variável a partir de uma classe, estamos criando um objeto, e a é apenas o tipo de objeto.

$C = $segundaConta;

$C = **Instância**
$segundaConta = **Objeto**

O objeto em si **não** guarda os valores na memória, ela tem um endereço que aponta para outro local e esse local tem todos os valores.
Sendo assim, **$C** e **$segundaConta apontam para o mesmo endereço.**

Uma **function** dentro de uma Class é chamado de **método**.

THIS:

$this → essa ex: $this→saldo;

$segundaConta → sacar(150)

$segundaConta = **Objeto**
$sacar = **método**
150 = **parâmetro**

$THIS se refere ao objeto que "chamou" o método.

Exemplo de function dentro da class:

```
public function sacar (float $valorASacar) {
        if( $valorASacar > $this → saldo) {
                echo "saldo insuficiente";
        } else {
                $this →saldo -= $valorASacar;
        }
}
```

Exemplo de function dentro da class (sem eles)

```
public function sacar (float $valorASacar) {
        if( $valorASacar > $this → saldo) {
        echo "saldo insuficiente";
        return;
        }
        $this → saldo -= $valorASacar;
}
```

ENCAPSULAMENTO:

Ajuda a garantir que os nossos projetos se mantenham em um estado consistente, e também facilita o uso da nossa classe por outros devs.

__CONSTRUCTOR:

Definir **__constructor ($parametros),** faz com que ao criar um **new conta (class)** faz com que ele seja executado como primeiro código, assim, colocando certas condições no código.

Exemplo:

```
public function __constructor ($cpfTitular, $nomeTitular) {
    $this → cpfTitular = $cpfTitular;
    $this → nomeTitular = $nomeTitular;
}
```

Para chamar o constructor:

```
$primeiraConta = new Conta (“123456789”, “Elbert”);
```

STATIC:

É usado para definir que um método ou atributo em uma classe é estático. Significa que aquele método/atributo pertence a classe e não a uma instância.

Exemplo:

```
private $numeroDeContas = 0;
```

__DESTRUCT:

É chamado quando não há mais nenhuma referência para o objeto em questão.
Destrói um atributo que não está mais endo utilizado na memória/que não foi atribuído a uma variável.

É possível que um objeto tenha outro objeto com valor de um de seus atributos. Isso é chamado de **composição.**

CURSO AVANÇADO DE ORIENTAÇÃO A OBJETOS COM PHP
HERANÇA, POLIMORFISMO E INTERFACES

EXTENDS →

Extende a classe para outra class

Exemplo:

Class Titular extends Pessoa

PILARES DA ORIENTAÇÃO A OBJETOS:

ABSTRAÇÃO → ENCAPSULAMENTO → HERANÇA → POLIMORFISMO

PARENT:

Reutiliza o construtor base em outra construct

Exemplo:

parent::__construct ($nome, $cpf);

PUBLIC, PROTECTED E PRIVATE:

PUBLIC perite que todos possam acessar o membro, inclusive fora da class onde foi definido, através de uma instância.
PROTECTED permite que a classe atual e as classes filhas tenham acesso.
PRIVATE permite apenas a classe atual tenha acesso.

HERANÇA:

Pode auxiliar com a duplicação de código. Para acessar um membro da classe base, pode ser utilizar **PARENT.**

NAMESPACE:

Trata-se de uma forma de organizar o código das classes e métodos, para evitar colisão destas classes devido a terem o mesmo nome.

Exemplo:

namespace pastaAtual \ subPasta;

Acessar o **namespace** com **use** para pode acessar a classe:

Exemplo:

require_once 'src/modelo/pessoa.php';

use modelo/pessoa;