

FIA/P GRADUAÇÃO

Digital Experience Platform

Prof. Andrey Masiero

#01 – Introdução ao PHP



ANDREY ARAUJO MASIERO

PROFESSOR GRADUAÇÃO E MBA NA FIAP

- MAIS DE 16 ANOS DE EXPERIÊNCIA
- MESTRADO EM ENGENHARIA ELÉTRICA COM ÊNFASE EM IA
- DOUTORADO EM ENGENHARIA ELÉTRICA COM ÊNFASE EM IA
- LECIONO TODOS OS NÍVEIS DE PROGRAMAÇÃO A 10 ANOS

CONTEÚDO: ANDREYMASIERO.COM

SEGUIE NO INSTA: @ANDREYMASIERO

LINKEDIN: ANDREYMASIERO

EMAIL: PROFANDREY.MASIERO@FIAP.COM.BR

GITHUB: AMASIERO

Objetivo da disciplina

- Preparar você para o mercado de trabalho.
- Apresentar duas das principais linguagens de programação na web PHP e JavaScript (NodeJS).
- Criar soluções utilizando as linguagens e banco NoSQL.
- Realizar modelagem dos sistemas através do paradigma orientado à objetos, utilizando essas linguagens.

Conteúdo Programático

- PHP + MongoDB – IDE PHPStorm
- NodeJS + MongoDB – IDE WebStorm

METODOLOGIA E AVALIAÇÕES

- **Metodologia**

- Aulas “Hands On”
- Projetos no Github (links nos slides de cada aula)
- Conteúdo extra no Google Classroom (Código de acesso à turma: **rifp4hl**)

- **Avaliações**

- **3 atividades individuais** – Prática
- **Case semestral** – Prática
- **AM** – breve num cinema perto de você! =D

PHP, linguagem de script é.

Yoda, Mestre Yoda.

O que é PHP?

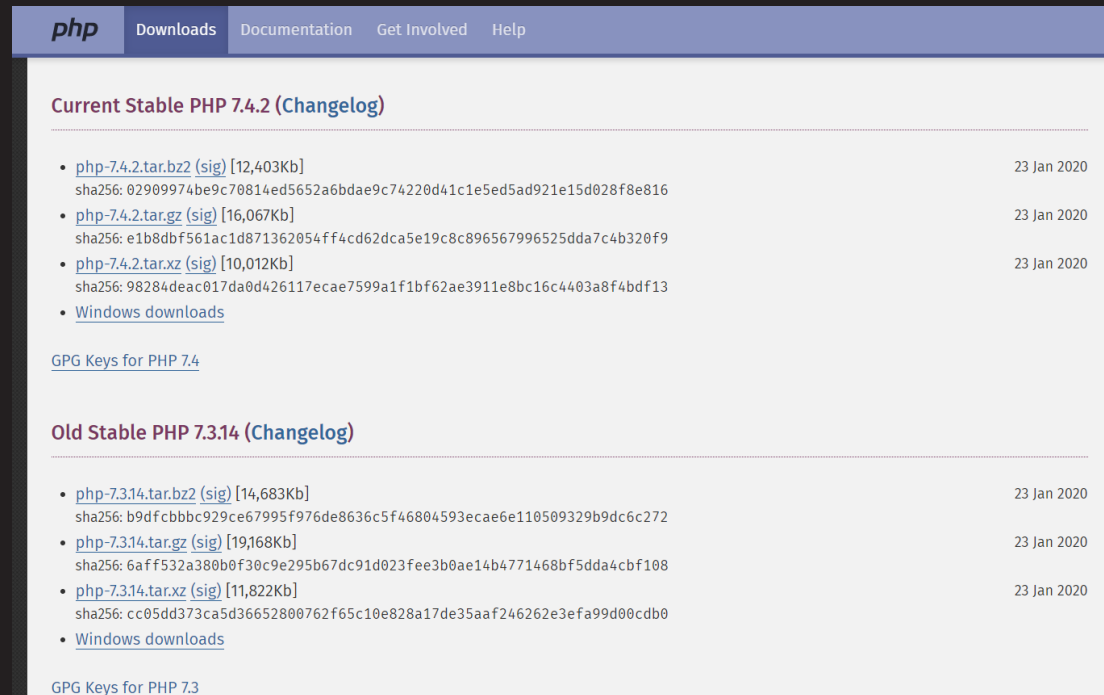
- Segundo o php.net:

PHP é uma linguagem de script de propósito geral, que possui um foco principal em desenvolvimento Web.

Rápido, flexível e pragmático, o PHP está presente em projetos como blogs, até os mais populares sites do mundo.

Versão Atual

- Sua versão atual é a 7.4.2;
- A versão anterior, também estável, era a 7.3.14.
- Para nós, neste curso, quaisquer versão 7 poderá ser utilizada.



The screenshot shows the PHP Downloads page. The navigation bar includes links for 'php', 'Downloads', 'Documentation', 'Get Involved', and 'Help'. The main content is divided into two sections: 'Current Stable PHP 7.4.2 (Changelog)' and 'Old Stable PHP 7.3.14 (Changelog)'. Each section lists download links for tar.bz2, tar.gz, and tar.xz archives, along with their SHA256 hashes and the date '23 Jan 2020'. There are also links for 'Windows downloads' and 'GPG Keys' for each version.

Current Stable PHP 7.4.2 (Changelog)

- [php-7.4.2.tar.bz2 \(sig\)](#) [12,403Kb] 23 Jan 2020
sha256: 02909974be9c70814ed5652a6bdae9c74220d41c1e5ed5ad921e15d028f8e816
- [php-7.4.2.tar.gz \(sig\)](#) [16,067Kb] 23 Jan 2020
sha256: e1b8dbf561ac1d871362054ff4cd62dca5e19c8c896567996525dda7c4b320f9
- [php-7.4.2.tar.xz \(sig\)](#) [10,012Kb] 23 Jan 2020
sha256: 98284deac017da0d426117ecae7599a1f1bf62ae3911e8bc16c4403a8f4bdf13
- [Windows downloads](#)

[GPG Keys for PHP 7.4](#)

Old Stable PHP 7.3.14 (Changelog)

- [php-7.3.14.tar.bz2 \(sig\)](#) [14,683Kb] 23 Jan 2020
sha256: b9dfcbbbc929ce67995f976de8636c5f46804593ecae6e110509329b9dc6c272
- [php-7.3.14.tar.gz \(sig\)](#) [19,168Kb] 23 Jan 2020
sha256: 6aff532a380b0f30c9e295b67dc91d023fee3b0ae14b4771468bf5dda4cbf108
- [php-7.3.14.tar.xz \(sig\)](#) [11,822Kb] 23 Jan 2020
sha256: cc05dd373ca5d36652800762f65c10e828a17de35aaf246262e3efa99d00cdb0
- [Windows downloads](#)

[GPG Keys for PHP 7.3](#)

Projetos com PHP

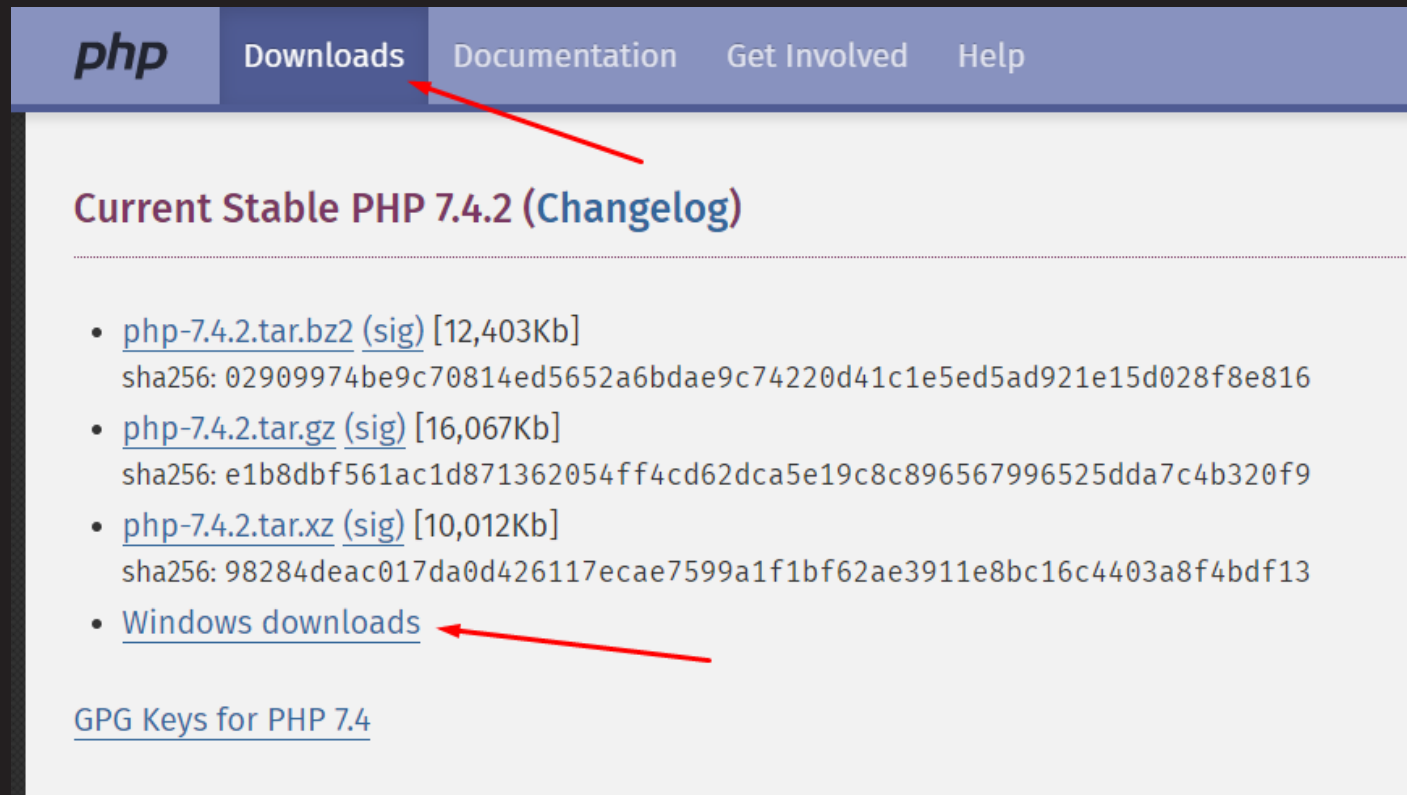
- IoT (Raspberry Pi + PHP) → <https://phppot.com/php/getting-started-with-iot-using-raspberry-pi-and-php/>
- GPS + IoT device + PHP (IBM) → <https://developer.ibm.com/tutorials/iot-php-app-iot-foundation-bluemix/>
- WordPress (Bloggging) → <https://br.wordpress.org/>
- Moodle (Educação) → https://moodle.org/?lang=pt_br

Instalando o PHP

Hora da mão na massa

Instalando PHP na sua máquina

- Entre na página de downloads e selecione Windows downloads da versão 7.4.2:



Instalando PHP na sua máquina

- Faça o download do arquivo ZIP:

PHP 7.4 (7.4.2)

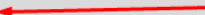
[Download source code](#) [23.15MB]

[Download tests package \(phpt\)](#) [13.3MB]

VC15 x64 Non Thread Safe (2020-Jan-21 22:49:25)

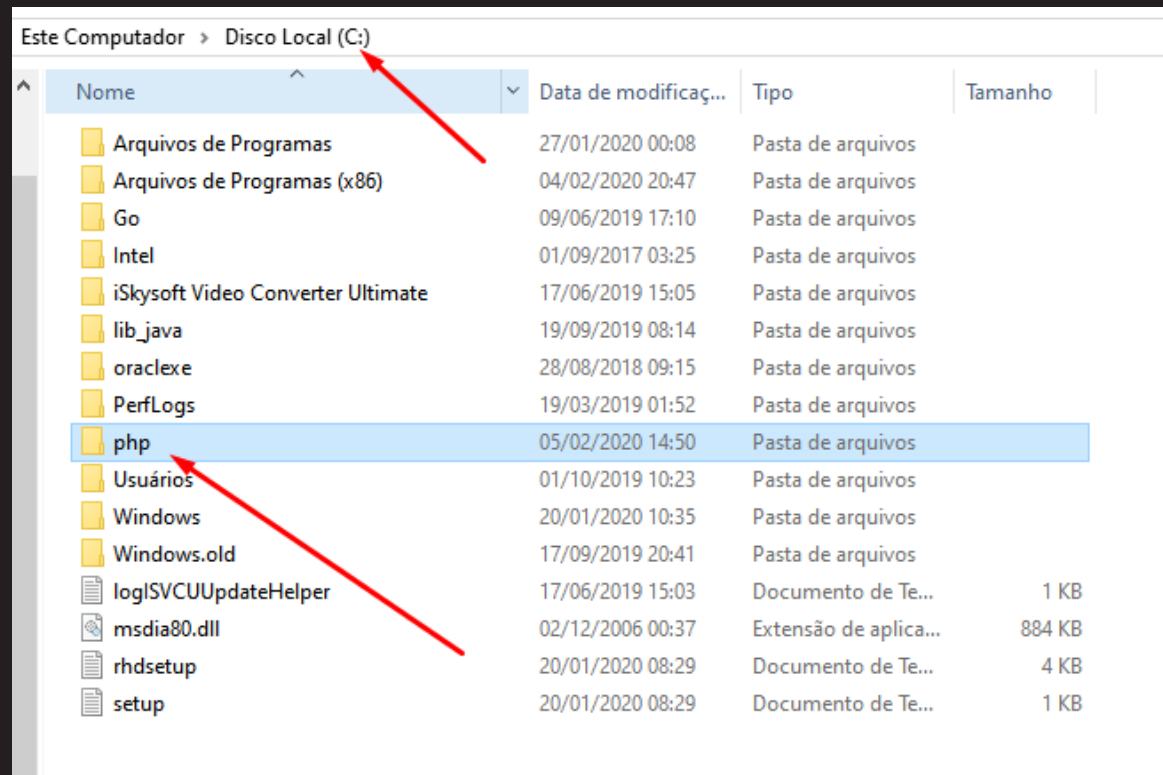
- [Zip](#) [24.75MB]
sha256: eee846d9bdc8baafe8f726a750433f3aaff8de7edd3a7918ca3dea4c99fdd1a4
- [Debug Pack](#) [21.87MB]
sha256: d184a13e4ff65311b6d944cf5ec3e1725cfbe0c3e9a0a771c331c6092e92497
- [Development package \(SDK to develop PHP extensions\)](#) [1.07MB]
sha256: e93eafedebabc0e6008043a0b6de5db60e6a47841871a8f91bfb94b92e8d1db1

VC15 x64 Thread Safe (2020-Jan-21 22:49:36)

- [Zip](#) [24.85MB] 
sha256: 034dc3ffb5583ee9e65f23ecae72ec93ecf1c31f694d1bdb7f90b81bc2b78897
- [Debug Pack](#) [21.87MB]
sha256: e06ed246deb825e7bcdfe342b607e9215c9c4abeb124d31720d412f6b801fa78
- [Development package \(SDK to develop PHP extensions\)](#) [1.07MB]
sha256: 1a4dea31343e0d378e4ef11a6afd3d8edf9eb819d5800e466a018e62afe48213

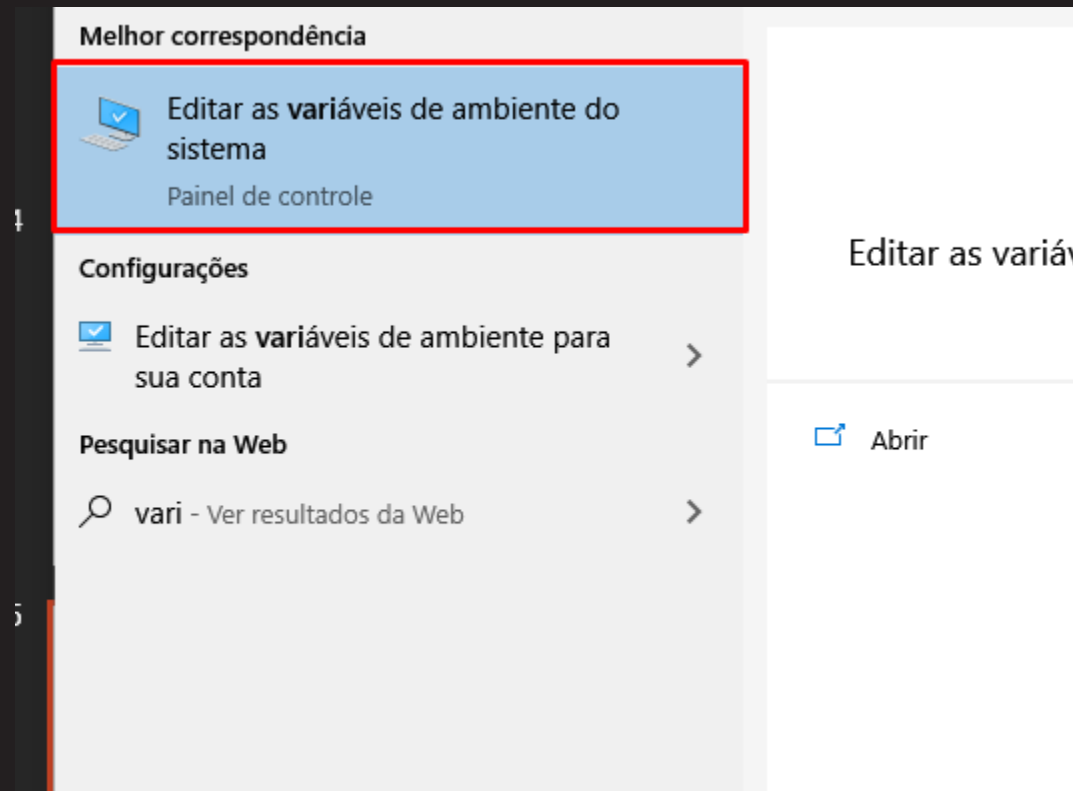
Instalando PHP na sua máquina

- Descompacte o arquivo e coloque em uma pasta fácil no seu computador, eu escolho normalmente a raiz (C:, no Windows).



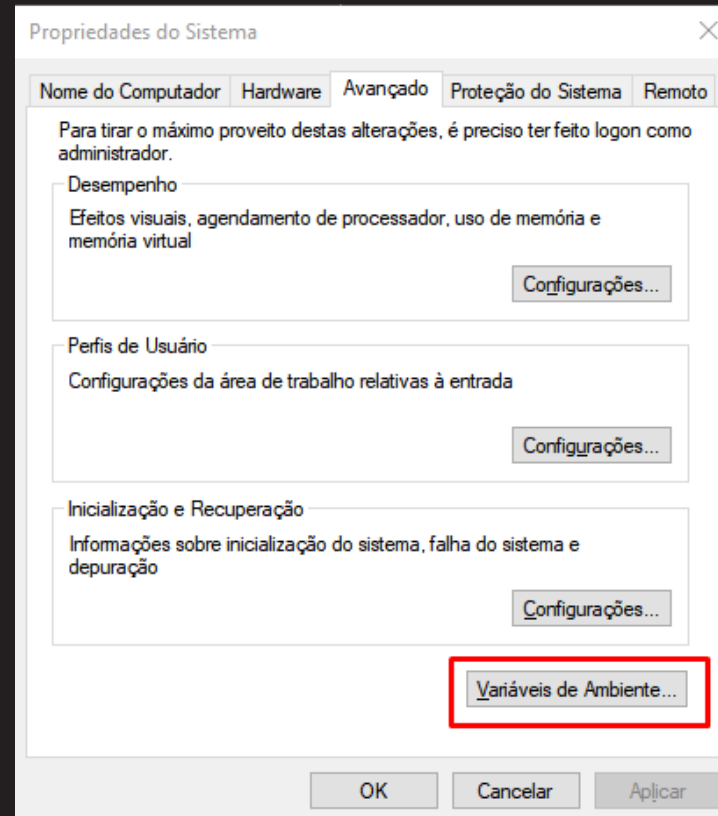
Instalando PHP na sua máquina

- No menu Iniciar, pesquise a palavra variáveis. Clique na opção selecionada:



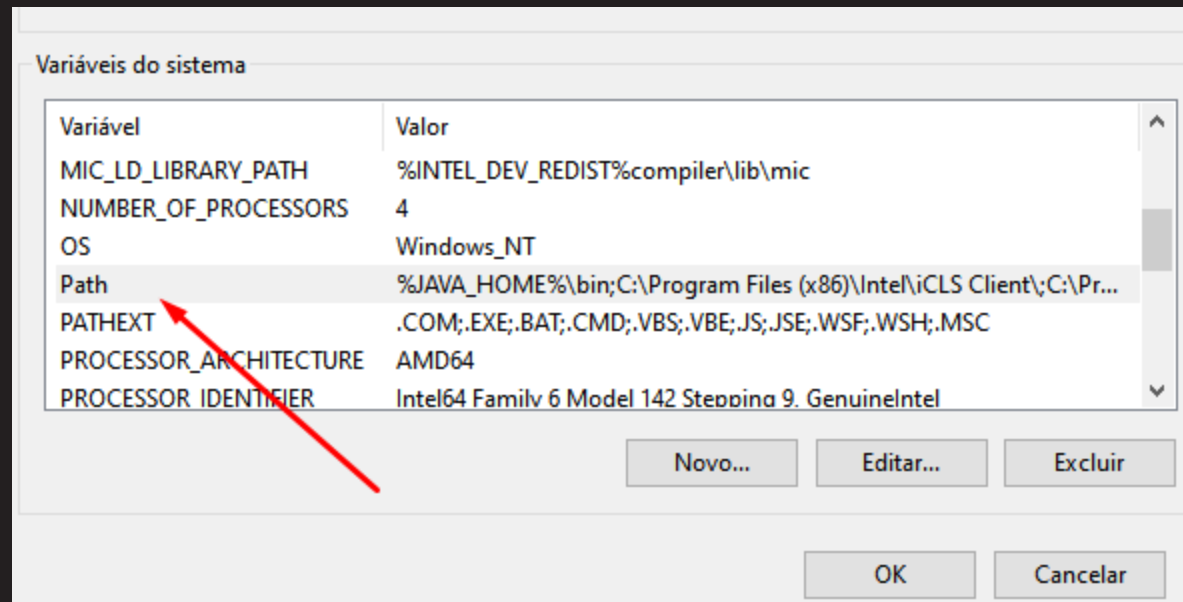
Instalando PHP na sua máquina

- Clique em Variáveis de Ambiente... Na janela que abriu:



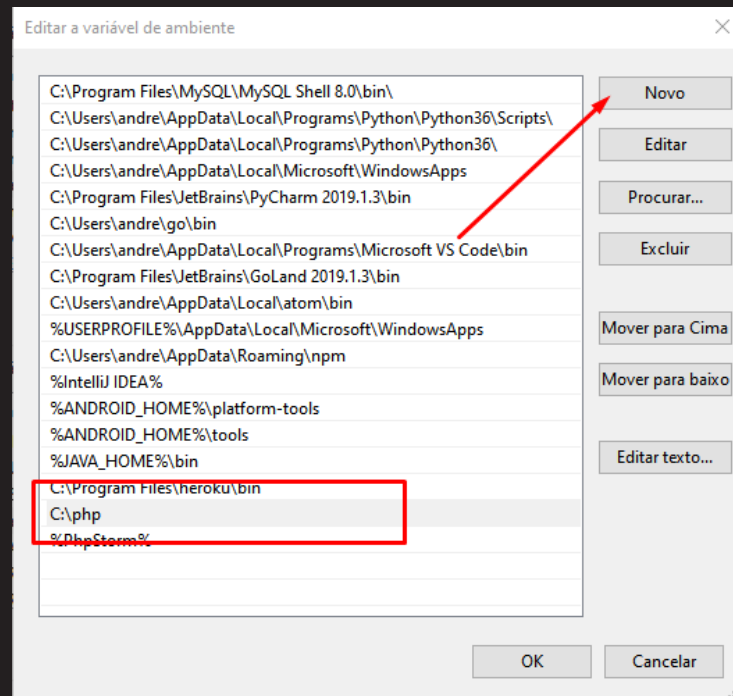
Instalando PHP na sua máquina

- Selecione a variável Path:



Instalando PHP na sua máquina

- Clique em Novo e adicione o caminho da pasta descompactada do php. Clique em OK e pronto. O PHP foi instalado.



Instalando em Linux com distribuição Debian based

- Vá no terminal e de o comando.

```
sudo apt-get install php
```

Princípios básicos do PHP

São os velhos amigos, variáveis, arrays, if else, for, while, foreach e por ai vai...

Olá mundo!

- Começando pelo clássico!

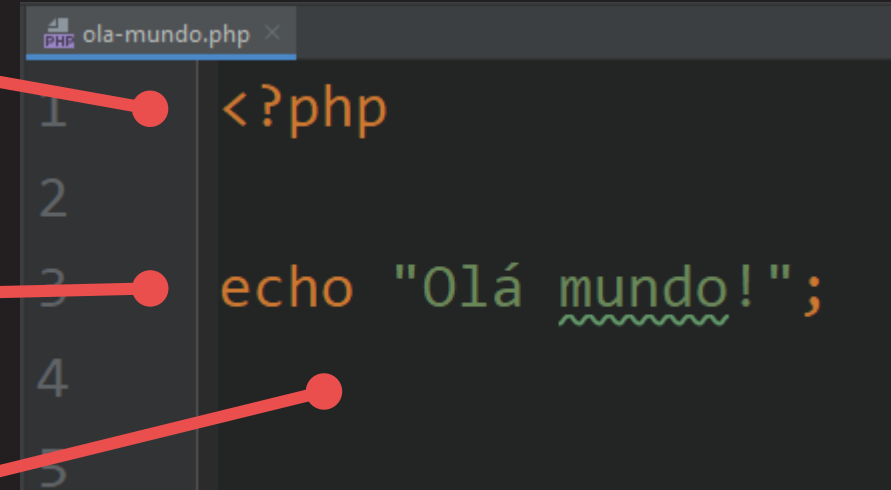
```
ola-mundo.php x
1  <?php
2
3  echo "Olá mundo!";
4
5
```



Olá mundo!

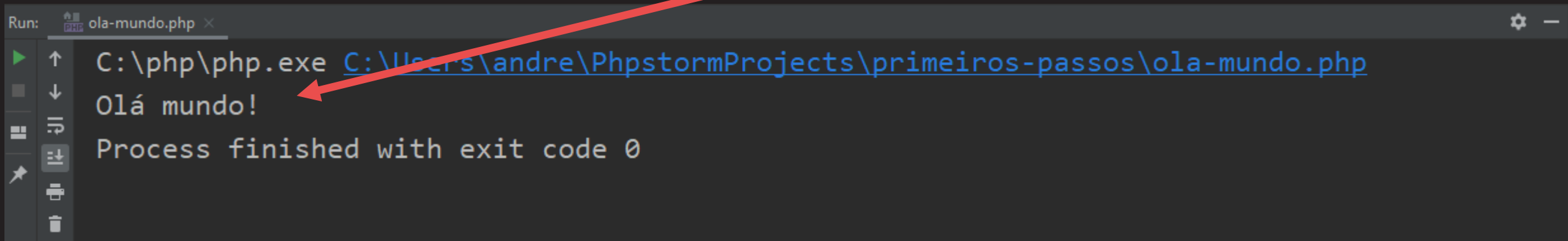
Informa que é um código PHP.

Comando para exibir um texto na tela.



```
1 <?php
2
3 echo "Olá mundo";
4
5
```

A screenshot of a code editor window titled 'ola-mundo.php'. It shows five lines of code. Red dots are placed at the end of lines 1, 3, and 4. Red arrows point from these dots to the text annotations on the left. The code on line 1 is '<?php', and the code on line 3 is 'echo "Olá mundo";'. The word 'mundo' is underlined in the original image.



```
Run: C:\php\php.exe C:\Users\andre\PhpstormProjects\primeiros-passos\ola-mundo.php
Olá mundo!
Process finished with exit code 0
```

A screenshot of a terminal window titled 'Run: ola-mundo.php'. It shows the command 'C:\php\php.exe C:\Users\andre\PhpstormProjects\primeiros-passos\ola-mundo.php' being executed. The output is 'Olá mundo!'. Below the output, it says 'Process finished with exit code 0'. A red arrow points from the text annotation 'Comando para exibir um texto na tela.' to the output 'Olá mundo!'.

Definindo variáveis

- Para definirmos uma variável em PHP, utilizamos o caractere \$ antes do nome:

```
1 <?php
2
3 $numero = 3;
4
```

- Exibir o número é fácil. Utilizamos o comando echo também:

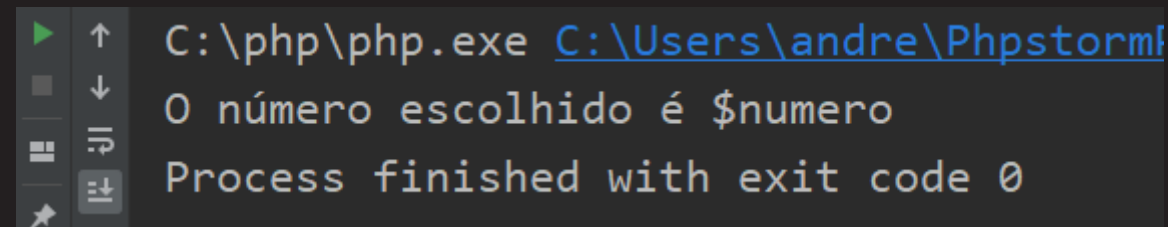
```
5 echo $numero;
```

Exibindo a variável em conjunto com texto

- O PHP aceita tanto aspas duplas quanto simples. Mas, qual a diferença? Vamos começar com as aspas simples.

```
echo 'O número escolhido é $numero';
```

- Confira o resultado:



```
C:\php\php.exe C:\Users\andre\Phpstormf
O número escolhido é $numero
Process finished with exit code 0
```

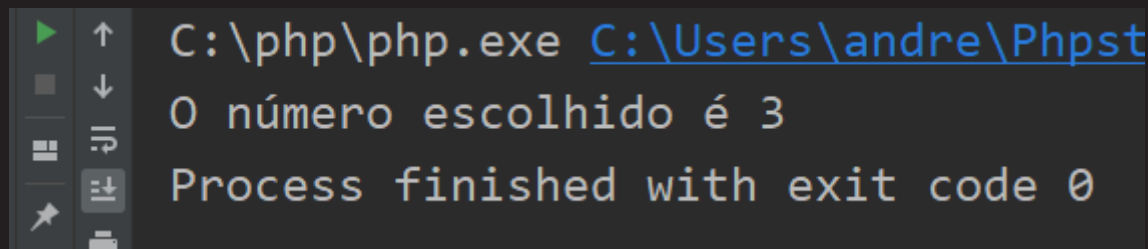
- Perceba que ele exibiu a variável como texto. Para o PHP aspas simples significa que ele não tem que se preocupar, só exibir direto tudo que está ali entre elas.

Exibindo a variável em conjunto com texto

- Então como fazer para exibir a variável? Bem, precisamos concatenar nossa string com a variável. O ponto (.) é o operador de concatenar.

```
echo 'O número escolhido é ' . $numero;
```

- Confira agora o resultado:



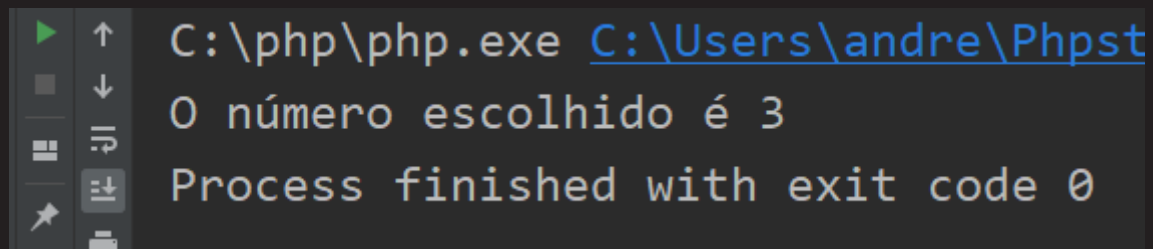
```
C:\php\php.exe C:\Users\andre\Phpst  
O número escolhido é 3  
Process finished with exit code 0
```

Exibindo a variável em conjunto com texto

- Mas, podemos facilitar isso utilizando as aspas duplas. Com elas o PHP faz um pré processamento antes de exibir a saída ao usuário.

```
echo "O número escolhido é $numero";
```

- Confira agora o resultado:



A screenshot of a terminal window with a dark background. The command prompt shows the execution of a PHP script: `C:\php\php.exe C:\Users\andre\Phpst`. The output of the script is displayed on the next line: `O número escolhido é 3`. The final line of the terminal output is `Process finished with exit code 0`. On the left side of the terminal, there is a vertical toolbar with icons for running, stopping, and debugging the process.

```
C:\php\php.exe C:\Users\andre\Phpst
O número escolhido é 3
Process finished with exit code 0
```

Jogo da Adivinhação

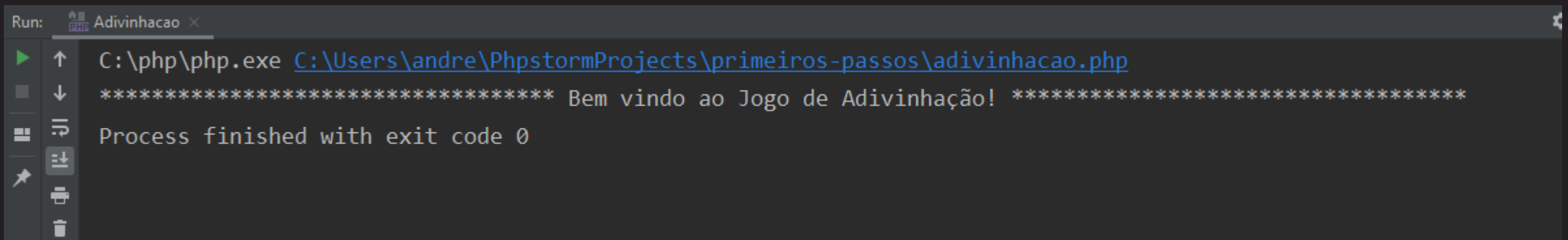
Verdade ou desafio!

Jogo da Adivinhação

- Vamos começar exibindo o cabeçalho. Utilizaremos o comando echo:

```
echo "*****";  
echo " Bem vindo ao Jogo de Adivinhação! ";  
echo "*****";
```

- Confira agora o resultado:



The screenshot shows a terminal window titled "Run: Adivinhacao x". The command executed is "C:\php\php.exe C:\Users\andre\PhpstormProjects\primeiros-passos\adivinhacao.php". The output is "***** Bem vindo ao Jogo de Adivinhação! *****". The terminal also shows "Process finished with exit code 0".

```
Run: Adivinhacao x  
C:\php\php.exe C:\Users\andre\PhpstormProjects\primeiros-passos\adivinhacao.php  
***** Bem vindo ao Jogo de Adivinhação! *****  
Process finished with exit code 0
```

Jogo da Adivinhação

- Saiu tudo na mesma linha! Mas por que? O PHP não quebra de linha automaticamente. Precisamos falar isso para ele. Existem duas maneiras de resolver isso. A primeira é utilizando o caractere especial `\n`, dentro da string em **aspas duplas**:

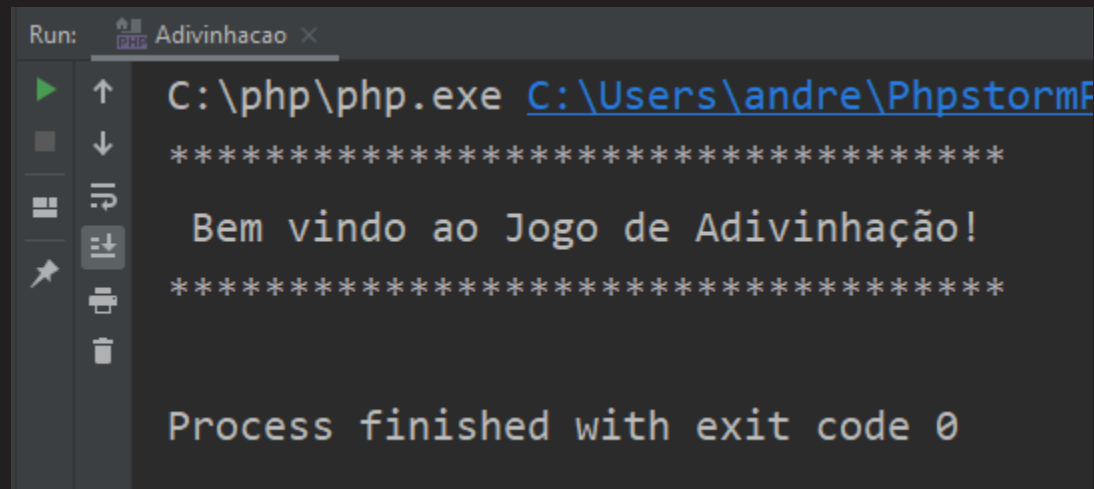
```
echo "*****\n";  
echo " Bem vindo ao Jogo de Adivinhação! \n";  
echo "*****\n";
```


Jogo da Adivinhação

- Aplicando o PHP_EOL:

```
echo "*****" . PHP_EOL;  
echo " Bem vindo ao Jogo de Adivinhação! " . PHP_EOL;  
echo "*****" . PHP_EOL;
```

- Confira o resultado:



The screenshot shows a terminal window titled "Run: Adivinhacao x". The command executed is "C:\php\php.exe C:\Users\andre\PhpstormF". The output displayed is:

```
*****  
  
 Bem vindo ao Jogo de Adivinhação!  
  
*****  
  
Process finished with exit code 0
```

Jogo da Adivinhação

- Agora vamos definir um numero secreto, e depois receber o um palpite do usuário, através da função readline:

```
$numeroSecreto = 3;  
  
$palpite = readline(prompt: "Digite seu número entre 1 e 100: ");  
  
echo "Você digitou: $palpite" . PHP_EOL;
```


Jogo da Adivinhação

- Resultado:

```
*****  
 Bem vindo ao Jogo de Adivinhação!  
*****  
Digite seu número entre 1 e 100: 10  
Você digitou: 10
```

Jogo da Adivinhação

- Próximo passo é comparar se o número secreto é igual ao digitado. Para isso, utilizaremos a instrução if:

```
if($palpite == $numeroSecreto) {  
    echo "Parabéns!\nVocê acertou o número secreto.";  
}  
  
echo "Fim de Jogo." . PHP_EOL;
```

Jogo da Adivinhação

- Vamos testar digitando 3, que é nosso número secreto, momentâneo.

```
*****  
 Bem vindo ao Jogo de Adivinhação!  
*****  
 Digite seu número entre 1 e 100: 3  
 Você digitou: 3  
 Fim de Jogo.
```



Jogo da Adivinhação

- O que aconteceu? Por que não disse que eu acertei? Vamos conferir o tipo das duas variáveis. Utilizaremos a função `gettype` nessa tarefa:

```
echo "palpite: " . gettype($palpite) . PHP_EOL;  
echo "numeroSecreto: " . gettype($numeroSecreto) . PHP_EOL;
```

- Confira o resultado:

```
palpite: string  
numeroSecreto: integer
```

Então é por isso, são
tipo diferentes!



Jogo da Adivinhação

- A função `readline` sempre retorna uma string. Para convertermos para um inteiro, utilizamos a função `intval`:

```
$palpite = intval(readline(prompt: "Digite seu número entre 1 e 100: "));
```

- Vamos ao resultado:

```
*****
 Bem vindo ao Jogo de Adivinhação!
*****
Digite seu número entre 1 e 100: 3
Você digitou: 3
Parabéns!
Você acertou o número secreto.Fim de Jogo.
```

Jogo da Adivinhação

- Mas podemos melhorar né? Que tal informar o usuário que ele perdeu o jogo. O if tem o completo de sua instrução, no caso o else:

```
if($palpite == $numeroSecreto) {  
    echo "Parabéns!\nVocê acertou o número secreto." . PHP_EOL;  
} else {  
    echo "Não foi dessa vez!\nTente novamente =D." . PHP_EOL;  
}
```

Jogo da Adivinhação

- Vamos conferir agora o caso negativo?

```
*****  
    Bem vindo ao Jogo de Adivinhação!  
*****  
Digite seu número entre 1 e 100: 8  
Você digitou: 8  
Não foi dessa vez!  
Tente novamente =D.  
Fim de Jogo.
```

Jogo da Adivinhação

- Mas poxa, trabalhar com o número fixo e exposto no código não é legal. Vamos deixar ele aleatório? Utilizaremos a função `random_int` que recebe o menor e o maior inteiro que pode existir.

```
$numeroSecreto = random_int(1, 100);
```

- Vamos exibi-lo apenas para conferir se está sendo gerado:

```
echo "Numero secreto é $numeroSecreto" . PHP_EOL;
```

Numero secreto é 66

Numero secreto é 40

Numero secreto é 46

Jogo da Adivinhação

- O número agora é gerado automaticamente e de maneira aleatória. Mas, fica muito difícil acertar de primeira esse número.
- Podemos fazer com que o usuário tenha um certo número de tentativas para trabalhar. Mas como podemos repetir a quantidade de tentativas que temos?
- Utilizaremos o laço de repetição **while**, para tal tarefa.

Jogo da Adivinhação

- O código fica assim:
- Definimos duas variáveis. Uma (tentativas) que representa o total de tentativas.
- A outra (tentativa) que representa a tentativa atual do jogador.

```
$tentativas = 3;
$tentativa = 1;

while($tentativa <= $tentativas) {
    $palpite = intval(
        readline(prompt: "Digite seu número entre 1 e 100: ")
    );
    echo "Você digitou: $palpite" . PHP_EOL;

    if($palpite == $numeroSecreto) {
        echo "Parabéns!\nVocê acertou o número secreto." . PHP_EOL;
    } else {
        echo "Não foi dessa vez!\nTente novamente =D." . PHP_EOL;
    }

    $tentativa = $tentativa + 1;
}
```

Jogo da Adivinhação

- Na sequência criamos o laço while comparando ambas variáveis criadas.
- Ao final do while, caso o jogador não tenha ganho, tentativa é incrementada em 1.

```
$tentativas = 3;
$tentativa = 1;

while($tentativa <= $tentativas) {
    $palpite = intval(
        readline(prompt: "Digite seu número entre 1 e 100: ")
    );
    echo "Você digitou: $palpite" . PHP_EOL;

    if($palpite == $numeroSecreto) {
        echo "Parabéns!\nVocê acertou o número secreto." . PHP_EOL;
    } else {
        echo "Não foi dessa vez!\nTente novamente =D." . PHP_EOL;
    }

    $tentativa = $tentativa + 1;
}
```

Jogo da Adivinhação

- Hora de testar!
- Show! Ele repetiu 3 vezes antes de finalizar o jogo.
- Vamos ao teste positivo.

```
Digite seu número entre 1 e 100: 94
Você digitou: 94
Não foi dessa vez!
Tente novamente =D.
Digite seu número entre 1 e 100: 74
Você digitou: 74
Não foi dessa vez!
Tente novamente =D.
Digite seu número entre 1 e 100: 25
Você digitou: 25
Não foi dessa vez!
Tente novamente =D.
Fim de Jogo.
```

Jogo da Adivinhação

- Roda o código!



- Eu acertei na segunda tentativa e ele continuou o jogo!
- O que aconteceu??????

```
Digite seu número entre 1 e 100: 10
Você digitou: 10
Não foi dessa vez!
Tente novamente =D.
```

```
Digite seu número entre 1 e 100: 69
Você digitou: 69
Parabéns!
Você acertou o número secreto.
```

```
Digite seu número entre 1 e 100: 100
Você digitou: 100
Não foi dessa vez!
Tente novamente =D.
Fim de Jogo.
```

Jogo da Adivinhação

- O problema é que o laço de repetição fica preso até que a condição seja falsa. Mas podemos interrompe-lo utilizando uma instrução chamada **break**:

```
if($palpite == $numeroSecreto) {  
    echo "Parabéns!\nVocê acertou o número secreto." . PHP_EOL;  
    break;  
} else {  
    echo "Não foi dessa vez!\nTente novamente =D." . PHP_EOL;  
}
```

- Pronto! Problema resolvido.

Jogo da Adivinhação

- Podemos deixar esse jogo mais interessante. O que acha de colocarmos níveis nele?
- Vamos fazer assim, o nível recruta terá 20 tentativas, o profissional 10 tentativas e o sobreviva 5 tentativas para descobrir o número secreto!
- Vamos criar as variáveis e então fazer um aninhamento de if e elses.

Jogo da Adivinhação

- Funciona!
- Mas nosso código não pode ficar melhor?
- Sempre!
- Vamos usar um truque com os if's

```
echo "Qual o nível de dificuldade?" . PHP_EOL;
echo "(1) Recruta (2) Profissional (3) Sobrevivente" . PHP_EOL;
$nível = readline( prompt: "Defina seu nível: ");

if($nível == 1) {
    $totalTentativas = 20;
} else {
    if($nível == 2) {
        $totalTentativas = 10;
    } else {
        if($nível == 3) {
            $totalTentativas = 5;
        }
    }
}
}
```


Jogo da Adivinhação

- Assim, ficou muito melhor!

```
echo "Qual o nível de dificuldade?" . PHP_EOL;
echo "(1) Recruta (2) Profissional (3) Sobrevivente" . PHP_EOL;
$nível = readline(prompt: "Defina seu nível: ");

if($nível == 1) {
    $totalTentativas = 20;
} elseif($nível == 2) {
    $totalTentativas = 10;
} elseif($nível == 3) {
    $totalTentativas = 5;
}
```

Jogo da Adivinhação

- Já que estamos refatorando nosso código, podemos mudar de laço de repetição, uma vez que determinamos o fim dele, sempre. Utilizaremos um for:

No próximo slide!

Jogo da Adivinhação

```
for($tentativa = 1; $tentativa <= $totalTentativas; $tentativa++) {  
    echo "Tentativa $tentativa de $totalTentativas" . PHP_EOL;  
  
    $palpite = intval(  
        readline(prompt: "Digite seu número entre 1 e 100: ")  
    );  
    echo "Você digitou: $palpite" . PHP_EOL;  
  
    if($palpite == $numeroSecreto) {  
        echo "Parabéns!\nVocê acertou o número secreto." . PHP_EOL;  
        break;  
    } else {  
        echo "Não foi dessa vez!\nTente novamente =D." . PHP_EOL;  
    }  
}
```

Jogo da Adivinhação

- Maravilha! Com o for ficou mais limpo o nosso código, sem tanta necessidades de diversas variáveis.
- Mas sempre podemos melhorar.
- Não existe nenhuma validação sobre o número inserido pelo jogador. Vamos fazer isso?

Jogo da Adivinhação

- A validação colocada após o palpite do jogador.

```
if($palpite < 1 && $palpite > 100) {  
    echo "Você deve digitar um número entre 1 e 100" . PHP_EOL;  
}
```

- Vamos testar!

```
Qual o nível de dificuldade?  
(1) Recruta (2) Profissional (3) Sobrevivente  
Defina seu nível: 1  
Tentativa 1 de 20  
Digite seu número entre 1 e 100: 102  
Você digitou: 102  
Não foi dessa vez!  
Tente novamente =D.  
Tentativa 2 de 20  
Digite seu número entre 1 e 100:
```

Jogo da Adivinhação

- Você percebeu que ele digitou errado, e ele continuou o código todo?
- Como podemos evitar isso? Bem, existe uma outra instrução que ajuda a pular a tentativa. Essa instrução é a **continue**.

```
if($palpite < 1 && $palpite > 100) {  
    echo "Você deve digitar um número entre 1 e 100" . PHP_EOL;  
    continue;  
}
```

Jogo da Adivinhação

- E agora, para fechar com chave de ouro, vamos dar umas dicas para o jogador a cada erro? Vamos criar as condições:

```
$acertou = $palpite == $numeroSecreto;  
$maior = $palpite > $numeroSecreto;  
$menor = $palpite < $numeroSecreto;
```

Jogo da Adivinhação

- A condição dentro do programa fica assim:

```
if($acertou) {  
    echo "Parabéns!\nVocê acertou o número secreto." . PHP_EOL;  
    break;  
} else {  
    echo "Errrrroooooouuuuu!!!!" . PHP_EOL;  
    if($maior) {  
        echo "O seu chute foi maior que o número secreto." . PHP_EOL;  
    } elseif($menor) {  
        echo "O seu chute foi maior que o número secreto." . PHP_EOL;  
    }  
}
```


Jogo da Adivinhação

- Masssssss..... Sempre podemos melhorar!
- Vamos colocar uma pontuação no jogo?
- Começaremos com 1000 pontos e vamos reduzir a cada tentativa errada.
- Para diminuir, vamos pegar a diferença entre o palpite e o numero secreto e subtrair dos pontos.

Jogo da Adivinhação

- Agora sim! Finalizamos...

```
if($acertou) {  
    echo "Parabéns!\nVocê acertou o número secreto." . PHP_EOL;  
    break;  
} else {  
    echo "Errrrroooooouuuuu!!!!" . PHP_EOL;  
    if($maior) {  
        echo "O seu chute foi maior que o número secreto." . PHP_EOL;  
    } elseif($menor) {  
        echo "O seu chute foi maior que o número secreto." . PHP_EOL;  
    }  
    $pontosPerdidos = abs( number: $palpite - $numeroSecreto);  
    $pontos = $pontos - $pontosPerdidos;  
}
```

Descanso

do Professor =D.

Exercícios

- Faça a lista de exercício disponível através link:
(necessário usuário do github)

<https://classroom.github.com/a/70Urtr0f>

Próximos Passos

O que veremos na próxima aula

Na próxima aula...

- Arrays, foreach, funções e PHP na Web



Copyright © 2020
Prof. Andrey Masiero

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

Do or do not. There is no try – Mestre Yoda