PHPの基本と演習

　これから学ぶために、最低限知っておくべきポイントをまとめます。 PHPは実践的な言語ですので、本授業において講義と演習を交互に行うことで、学習効率の向上を図ります。

1. PHPの基本構造

PHPコードは <?php ... ?> で囲み、HTMLに埋め込んで使います。

```
<!DOCTYPE html>
<html>
<body>
    <?php
        echo “Hello World!”;    // 画面に文字を出力
        print “Hello World!”;    // 画面に文字を出力
    ?>
</body>
</html>
```

ポイント:

echo は画面に表示する関数

// はコメント（実行されない）

2. 変数の使い方

変数は $ で始めます。型宣言は不要（PHPは動的型付け）。

```
$name = "太郎";  // 文字列

$age = 25;      // 整数

$price = 99.99; // 浮動小数点数

$is_active = true; // 真偽値
```

3. 主要なデータ型

型	例	説明
`string`	`"PHP"`	文字列
`int`	`100`	整数
`float`	`3.14`	小数
`bool`	`true` / `false`	真偽値
`array`	`[1, 2, 3]`	配列

```
$a = 10; $b = "text"; var_dump($a, $b);
```

3. 主要なデータ型　PHPのvar_dump()関数の用法

var_dump()はPHPのデバッグ用関数で、変数や式に関する詳細な情報を出力します。

よく使われる場面

- 変数に何が入っているか確認したいとき
- 関数の戻り値を検査したいとき
- データ構造を確認したいとき
- デバッグ時に値の変化を追跡したいとき

```
$a = 10;
$b = "text";
var_dump($a, $b);
```

4.主要な演算子　算術演算子

演算子	説明	例
`+`	加算	`$a + $b`
`-`	減算	`$a - $b`
`*`	乗算	`$a * $b`
`/`	除算	`$a / $b`
`%`	剰余（モジュロ）	`$a % $b`
`**`	累乗	`$a ** $b`
`-`	符号反転	`-$a`

演算子には優先順位があり、必要に応じて括弧 () を使って明示的に優先順位を指定することができます。

4.主要な演算子 代入演算子

演算子	説明	例
`=`	代入	`$a = $b`
`+=`	加算代入	`$a += $b`
`-=`	減算代入	`$a -= $b`
`*=`	乗算代入	`$a *= $b`
`/=`	除算代入	`$a /= $b`
`%=`	剰余代入	`$a %= $b`
`**=`	累乗代入	`$a **= $b`
`.=`	文字列結合代入	`$a .= $b`

4.主要な演算子 比較演算子

演算子	説明	例
`==`	等しい（値のみ比較）	`$a == $b`
`===`	等しい（値と型を比較）	`$a === $b`
`!=`	等しくない	`$a != $b`
`<>`	等しくない	`$a <> $b`
`!==`	等しくない（値または型）	`$a !== $b`
`<`	より小さい	`$a < $b`
`>`	より大きい	`$a > $b`
`<=`	以下	`$a <= $b`
`>=`	以上	`$a >= $b`
`<=>`	宇宙船演算子（PHP7以降）	`$a <=> $b`

4.主要な演算子 比較演算子(＝＝＝)の使い方

PHP の === は「厳密等価演算子」と呼ばれ、値とデータ型の両方が等しい場合に true を返します。

=== の基本的な特徴

- 値とデータ型の両方を比較します
- 型変換を行いません
- ==（緩やかな比較）よりも厳密な比較を行います

使用例

```
var_dump(5 === "5");  // false（型が異なる）
var_dump(5 === 5);    // true
```

厳密な比較が必要なアルゴリズム実装時=== を使用することで、予期せぬ型変換によるバグを防ぐことができます。PHPでは特に型が柔軟なため、=== を使うことが推奨される場面が多くあります。

4.主要な演算子 宇宙船演算子(<=>)の意味と使い方

- 宇宙船演算子（Spaceship Operator）はPHP 7で導入された比較演算子で、<=> という記号で表されます。この演算子は2つの値を比較し、大小関係に基づいて -1、0、1 のいずれかの整数値を返します。
- 数値、文字列、配列の比較、ソートなどに使えます。
- 利用例：

```
echo "a" <=> "a";  // 0
echo "a" <=> "b";  // -1
echo "b" <=> "a";  // 1
```

4.主要な演算子 論理演算子

演算子	名称	動作説明	例
`&&`	論理積	両方の条件がtrueの場合にtrue	`$a > 0 && $b < 10`
`||`	論理和	どちらかの条件がtrueの場合にtrue	`$x == 5 || $y == 3`
`!`	否定	条件の真偽を反転	`!($a == $b)`
`and`	論理積	`&&` と同じだが優先順位が低い [1]	`$a > 0 and $b < 10`
`or`	論理和	`||` と同じだが優先順位が低い [1]	`$x == 5 or $y == 3`
`xor`	排他的論理和	どちらか一方のみがtrueの場合にtrue（両方trueは不可） [1] [8]	`$a xor $b`

&&/and：左辺がfalseの場合、右辺を評価しない
||/or：左辺がtrueの場合、右辺を評価しない

優先順位の違い

&& > || > and > or

4.主要な演算子 インクリメント/デクリメント演算子

演算子	説明	例
`++`	インクリメント	`++$a`
`++`	インクリメント	`$a++`
`--`	デクリメント	`--$a`
`--`	デクリメント	`$a--`

4.主要な演算子　文字列演算子

演算子	説明	例
`.`	文字列結合	`$a . $b`
`.=`	文字列結合代入	`$a .= $b`

4.主要な演算子 配列演算子

演算子	説明	例
`+`	配列の結合	`$a + $b`
`==`	配列の等価比較	`$a == $b`
`===`	配列の同一比較	`$a === $b`
`!=`	配列の不等価比較	`$a != $b`
`<>`	配列の不等価比較	`$a <> $b`
`!==`	配列の非同一比較	`$a !== $b`

4.主要な演算子 ビット演算子

演算子	説明	例		
`&`	ビットAND	`$a & $b`		
`	`	ビットOR	`$a	$b`
`^`	ビットXOR	`$a ^ $b`		
`~`	ビットNOT	`~$a`		
`<<`	左シフト	`$a << $b`		
`>>`	右シフト	`$a >> $b`		

4.主要な演算子 その他の演算子

演算子	説明	例
`?:`	三項演算子	`$a ? $b : $c`
`??`	Null合体演算子（PHP7以降）	`$a ?? $b`
`<=>`	宇宙船演算子（PHP7以降）	`$a <=> $b`
`->`	オブジェクトのメンバアクセス	`$obj->property`
`::`	クラスの静的メンバアクセス	`Class::method()`
`instanceof`	オブジェクトのクラスチェック	`$a instanceof B`

5. 配列の操作

```
// 配列の作成
$fruits = ["apple", "banana", "orange"];

// 値の取得
echo $fruits[0]; // "apple"

// 連想配列（キーと値）
$user = [
    "name" => "田中",
    "age" => 30
];
echo $user["name"]; // "田中"
```

6. 条件分岐(if文、if...elseif...else文)

- **単純なif文**

```
if (条件式) {
    // 条件が真(true)の場合に実行されるコード
}
```

- **if-else文**

```
if (条件式) {
    // 条件が真(true)の場合
} else {
    // 条件が偽(false)の場合
}
```

- **if-elseif-else文**

```
if (条件式1) {
    // 条件式1が真の場合
} elseif (条件式2) {
    // 条件式2が真の場合
} else {
    // すべての条件が偽の場合
}
```

6. 条件分岐(if文、if...elseif...else文)

```
$score = 80;

if ($score >= 60) {
    echo "合格！";
} else {
    echo "不合格...";
}
```

```
$t = date("H");
if ($t < "10") {
  print “おはよう!";
} elseif ($t < “18") {
  print “こんにちは!";
} else {
  print “こんばんは!";
}
```

7. 複数の条件分岐 switch文

```
switch ($variable) {
    case value1:
        // $variableがvalue1に等しい場合の処理
        break;
    case value2:
        // $variableがvalue2に等しい場合の処理
        break;
    default:
        // どのcaseにも該当しない場合の処理
        break;
}
```

7. 複数の条件分岐 switch文

```
$day = "月曜日";

switch ($day) {
    case "月曜日":
        echo "今日は月曜日です。";
        break;
    case "火曜日":
        echo "今日は火曜日です。";
        break;
    case "水曜日":
    case "木曜日":
        echo “今日は水曜日か木曜日です。";
        break;
    default:
        echo “今日は金曜日、土曜日、または日曜日です。";
}
```

8. ループ処理　forループ

```
for (初期化式; 条件式; 更新式) {
    // 繰り返し実行する処理
}
```

```
for ($i = 1; $i <= 5; $i++) {
    echo $i . "回目<br>";
}
```

8. ループ処理　foreachループ（配列用）

配列の値を処理する場合

```
foreach ($array as $value) {
    // $valueを使用した処理
}

$colors = ["red", "blue", "green"];
foreach ($colors as $color) {
    echo $color . "<br>";
}
```

8. ループ処理　foreachループ（配列用）

キーと値の両方を処理する場合

```
$person = [
    "name" => "山田太郎",
    "age" => 25,
    "email" => "taro@example.com"
];

foreach ($person as $key => $value) {
    echo "$key: $value<br>";
}
```

9. ループ処理　while文

```
while (条件式) {
    // 条件がtrueの間、繰り返し実行する処理
}

$i = 1;
while ($i <= 5) {
    echo $i . "<br>";
    $i++;
}
```

10. 関数の定義

```
function add($a, $b) {
    return $a + $b;
}

$result = add(3, 5); // 8
```

PHPの変数は、使える有効範囲が限られており、この範囲のことをスコープ(scope)と言います。

{ }で囲まれた関数内で宣言した変数のことをローカル変数、その外で宣言した変数をグローバル変数と言います。そして、それぞれの変数の効力が及ぶ有効範囲をローカルスコープ、グローバルスコープと呼びます。

基本的に、ローカルスコープは{ }で囲まれた関数内を指しますが、グローバルスコープはプログラム全体どこでも使うことができます

11. フォーム処理の基本

```
<!-- HTMLフォーム -->
<form method="POST" action="submit.php">
    <input type="text" name="username">
    <button type="submit">送信</button>
</form>
```

```
// submit.php
$username = $_POST["username"]; // フォームデータの取得
echo $username . "さん、こんにちは！";
```

12. データベース接続（MySQL例）

```
$conn = new mysqli("localhost", "user", "password", "db_name");

// 接続チェック
if ($conn->connect_error) {
    die("接続失敗: " . $conn->connect_error);
}

// クエリ実行
$result = $conn->query("SELECT * FROM users");
```

演習の概要

- HTML5とPHPプログラミングでページを作る
- 変数と配列を操作し、簡単なプログラミングを行う
- if や for で制御フローを理解し、簡単なプログラミングを行う

- GUIを習い簡単なフォーム処理を作ってみる