PHP言語　初級

Field-UP
～技術とヒトに寄り添う～

Contents

- ■本日の講座の目的と皆さんの思いの言語化
- ■PHPではどんなことができるの?
- ■ライブラリってなに？
- ■考えることを楽しもう
- ■具体的なコードの書き方
- ■本日のまとめと感想

本日の講座の目的と皆さんの思いの言語化

スキルを身につけるときや覚えるときには目的意識を持つことがとても重要になってきます

下のことを言語化しましょう

- いま具体的にどんなことが課題になっていますか？
- それを解決するために今は、何が必要だと思っていますか？
- 今日は何を（どんなスキル）持ち帰りたいですか？

PHPではどんなことができるの?

言語の特徴

PHPという言語の特徴をまとめながら見ていきましょう

WEBアプリケーション開発特化

一般的なWEBサイトは静的コンテンツといい、何かを表示するだけや、ページの遷移のみのことしかできません
しかし、PHPを使うことによってユーザーの操作によって、ページ内で検索や、表示を変更したりと動的なコンテンツを作成することができます

手軽なプログラミング

最大の特徴は手軽さです
HTMLのコードに直接組み込めて実行することができます
フレームワークを必要としないプログラミング言語なので、コーダーの好きに機能をコーディングできます

サービスとして多く展開

PHPやWordpressといったCMS(ｺﾝﾃﾝﾂ・ﾏﾈｼﾞﾒﾝﾄ・ｼｽﾃﾑ)や、メルカリ、Yahoo!JAPANとよく知られている上に多く利用されているサービスの開発言語になっています
それだけ世の中に根付いている言語だということがわかります

ライブラリってなに？

違いがわかれば覚えやすい

ライブラリとは、**既にいろんな機能が書かれたプログラム**のようなものです

これを使うことによって、**ゼロからプログラムを組まなくてもいい**ことがあります

ライブラリだけで考えてしまうとわかりづらいと思うので、他のことと併せて覚えていきましょう

下の図のように覚えることがオススメです

モジュール

拡張子が「.php」のファイル
単体でも動く

コンポーネント

拡張子が「.php」のファイル
単体では動かない

パッケージ

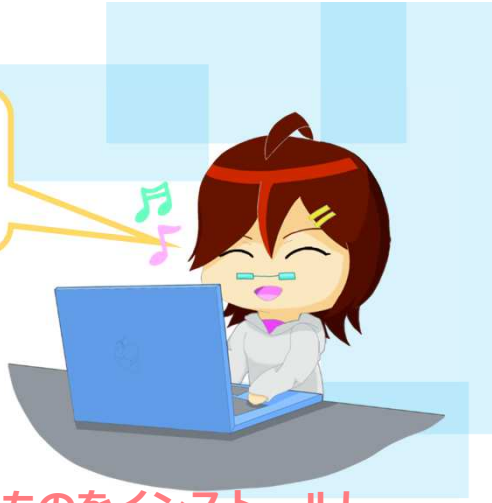
モジュールの集まったもの

ライブラリ

複数の機能のまとめられた
プログラム
単体では動かない

ライブラリってなに？

よく使われるライブラリ



PHPのライブラリを使えるようにするためには二つのステップがあります

①ライブラリ本体をインストールする

②ライブラリを実行するために必要な環境をインストールする

PHPのライブラリは、**一つのみでは機能しないものもあり、Composerというものをインストールしていくことによって、使いたいライブラリに付随するライブラリも同時にインストール**してくれます

ライブラリ名	使用方法
PHP Image Upload Class	画像投稿
Ratchet	チャット制作
Upload	ファイルのアップロード
Sentinel	ユーザー登録
Guzzle バージョン6	API開発
Propel	DB管理補助
Two Factor Auth	二段階承認
PHP Debug Bar	デバッグ
Stripe	決済サービス

***Composerのインストール方法は別のページまとめてます**

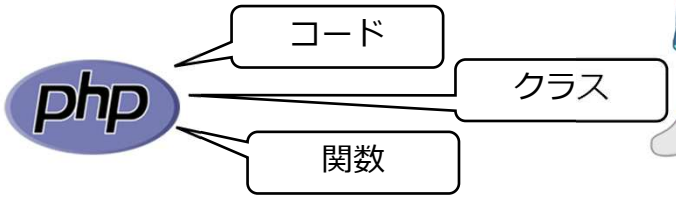
考えることを楽しもう

オブジェクト、クラス、インスタンス

ここでもそれぞれの違いから見ていきましょう

オブジェクト

プログラムに書かれているすべてのモノや事柄のこと



クラス

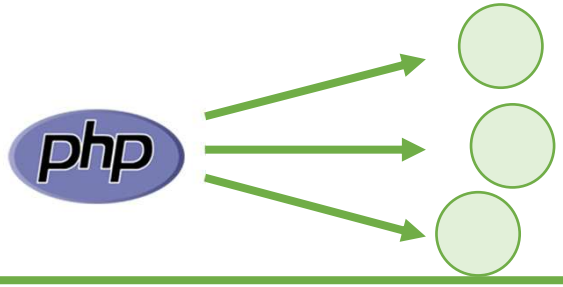
設計図のようなもの
プロパティ(特徴)と**メソッド(機能)**が書かれている

~設計図1~

・こんな形
・こんな特徴
・こんな動きをする

インスタンス

設計図を元に実際に作ったもの
これがあるから大量に同じものを作れる



考えることを楽しもう

どうやって考えればいいの？

一から何かを作るのは、慣れてきた人でも難しいものです

大事なこと下のようなことです

- **これから、または、これはどんなアルゴリズムなのかを理解する**
- **人が実際に行う動作としては、どんな感じになるのだろうか？**
- **考えるための材料は集められたか？**

プログラムはあくまで、**人のおこなう動作を勝手にやってくれるものにすぎません**

人のおこなっている動作を超高速で、かつ、正確におこなっているだけです

考えることを楽しもう

考えるための材料集め

プログラムを作るためには、先ずはアルゴリズムを考える必要があります
そのポイントは下のようになります

- **何をしたいのか(目的)**
- **どんな状態からスタートするのか(起動条件)**
- **何に対して操作をしたいのか(セルなのか、中の文字なのか)**
- **どんな処理(動作・計算)をしてほしいのか**
- **人がどんな茶々を加える可能性があるのか(バグを潰す)**

上記のポイントを考えながら、必要に応じてフローチャートを作ってまとめていきましょう
＊ボールペンと紙でサラサラっと書くことをお勧めします
　間違えたらそのまま残して、別のスペースに書きましょう

考えることを楽しもう

フローチャートの作り方のコツ

フローチャート作るための一番のポイントは**規則性**です
その作業の規則性をいかに見つけるかが重要です

＊この作業は同じことの繰り返しなんだよな～

＊でもこの作業の時は違うんだよな～

＊ケースが多すぎなんだけど、ある程度規則性はあるんだよな～

先ずは、こんなことが見つかってこれば大丈夫です!!
ココから細かく落とし込んでいけばいいだけのことです
アルゴリズムは大きく分けると、3つのブロックにしか分かれません

入力操作

処理(加工)操作

出力操作

考えることを楽しもう

フローチャートの作り方のコツ

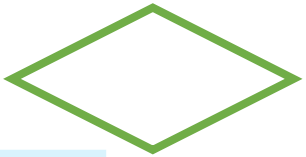
言語化した作業内容をフローチャートにしていきます

フローチャートの形をご説明します

開始、終わり

処理、作業

読み込み、書き出し、出力、入力



条件分岐

繰り返し処理

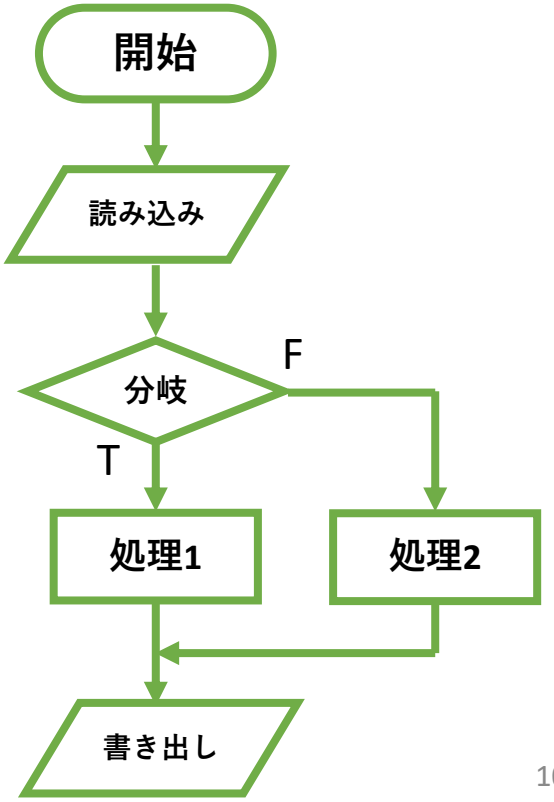
ワンポイント

フローチャートを作る目的は、プログラムを作るためなので、ココではあまりや形などにはこだわらないようにしましょう

大事なのは、**作業をどこで区切っていけばいいか**です

先ずは**大雑把に区切って**いきながら後から細かく分けていきましょう

参考例



考えることを楽しもう

フローチャートの作り方のコツ

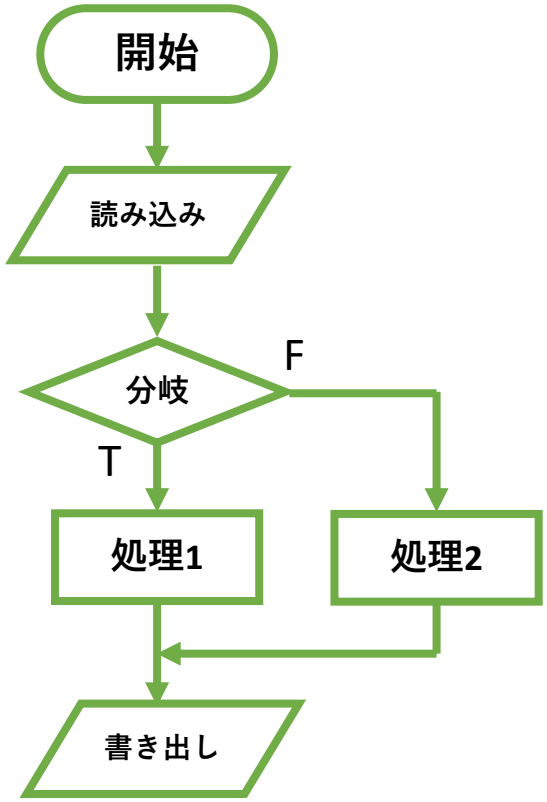
作業の言語化をさらに細分化すれば、あとはそれをコードにするだけです

大事なのは言語化です!!

言語化 → チャート化 → コード化

・A1セルの値を読み取る
・読み込んだ値が、4以上なら
　処理1をして、違うなら処理2
　とをする
・処理1は、4に5をかける
・処理2は、4に5を足す
・結果をB3に書き出す

・
・
・




```
t = Cells(1,1)
If t >= 4 then t = t * 5
      else t = t + 5
End If
Cells(3,2) = t
```

・
・
・

Excel操作の例です

考えることを楽しもう

変数とはなにか？

変数やコード1行の見方、考え方を説明します

左辺 = 右辺

上記のようなコードを読み下すと…

右辺の内容を左辺に反映させなさい(入れなさい)

というように読みます

例題

左辺の変数には何が入っていることになるでしょう？

① $x = 1;

② $y = “ストアカ”;

③ $z = 2 + 5;

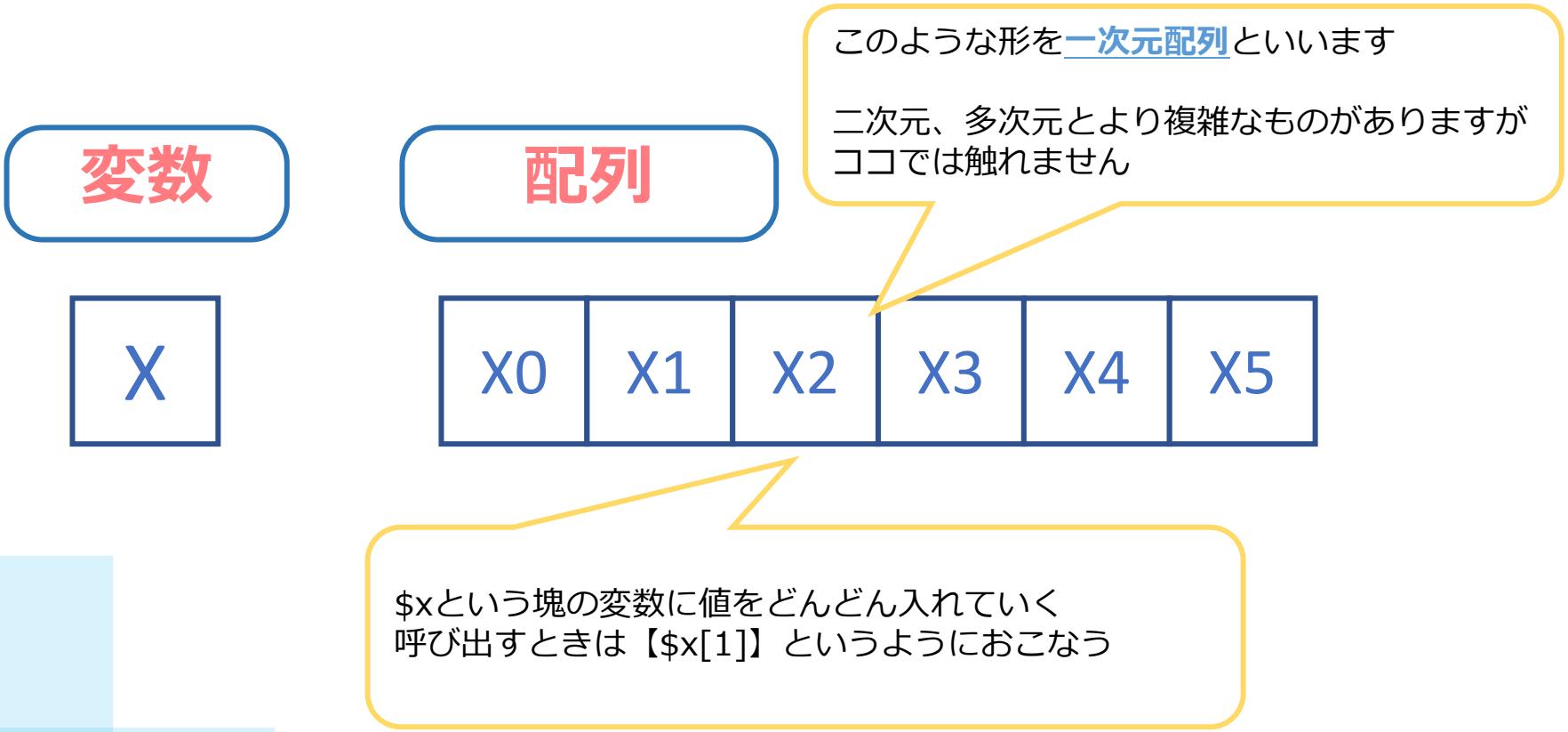
考えることを楽しもう

配列とは何か？

ここから少しずつ難易度が上がってきます

配列は、簡単に言うと変数の塊です

変数は、一つの値しか入れられませんが、配列は複数の値を順番に並べて入れることができます



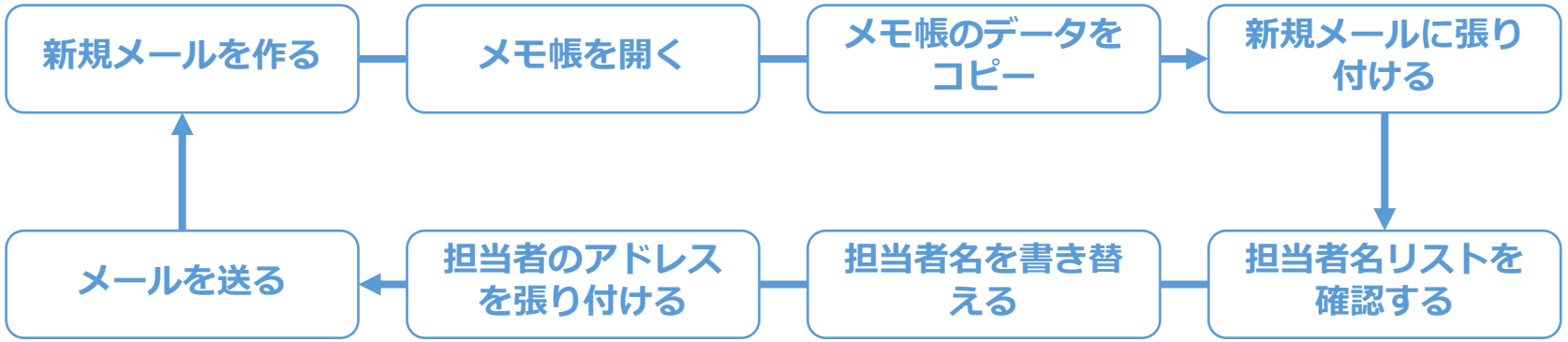
考えることを楽しもう

繰り返し処理とは何か？

作業するときに同じようなことを何度もした経験はないですか？

例えば、メモ帳のデータを担当者名を変えて、メールソフトに貼り付けて送信するといったような作業です

こんな時には繰り返し処理を使うと便利です



繰り返しは、回数で終わったり、何かがなくなるまで続けるということが多いと思います
＊今回の場合は、担当者名をすべて読み終わるまでなど

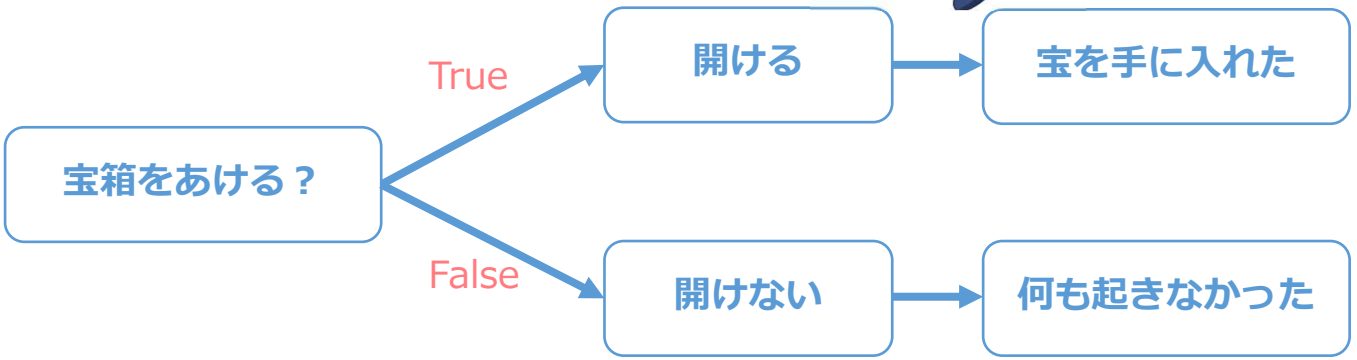
考えることを楽しもう

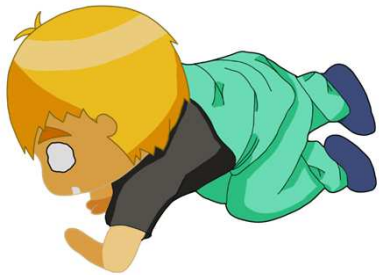
条件分岐とはなに？

難しく書いていますが、YESかNOかを判断することです

プログラミングではTrueかFalseといいます

これがあるからプログラムは様々な動作をすることができます





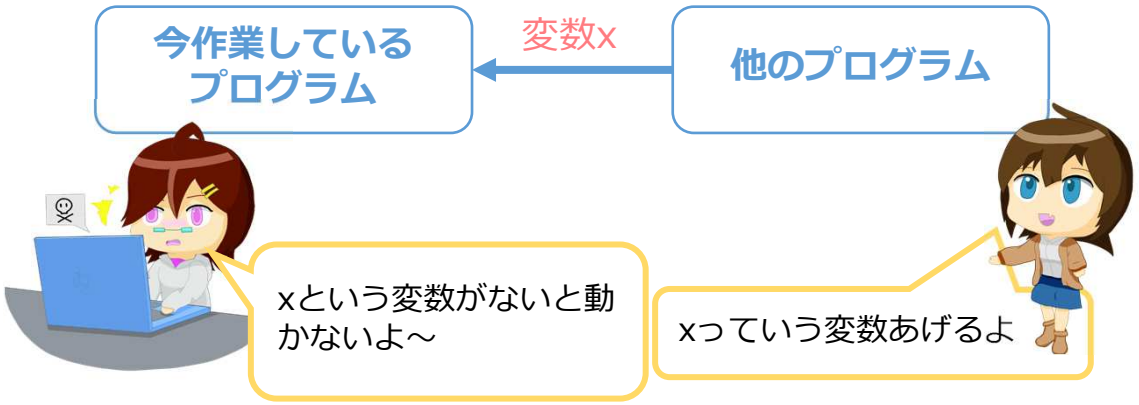
考えることを楽しもう

引数と戻値って何？

日常生活ではほぼ触れない言葉だと思います

これは値や変数をプログラムがどう扱っているかを表している言葉です

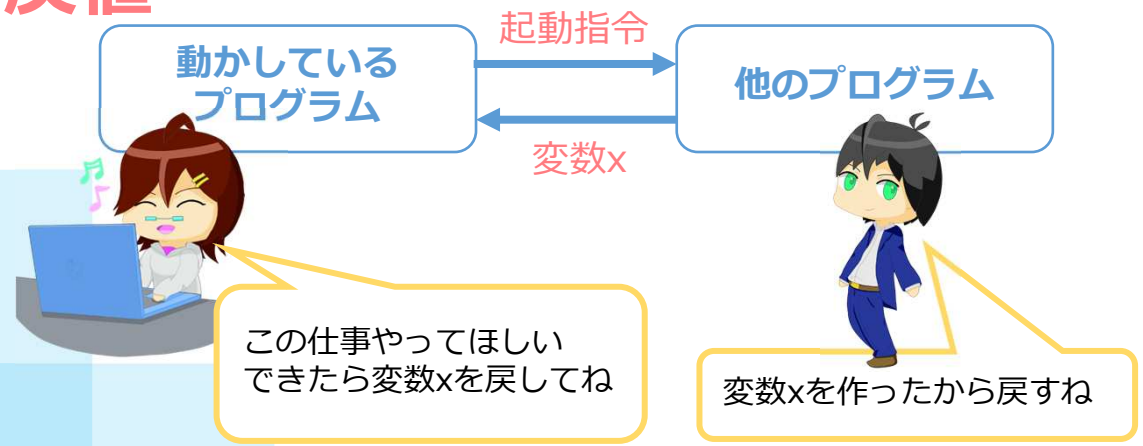
引数



~解説~

他のプログラムから**変数や値を引き込みますよ**という意味で、引数といわれています

戻値



~解説~

プログラムで他のプログラムを起動させて、その動かしたプログラムで作った**変数や値を元のプログラムに戻しますよ**という意味で、引数といわれています

考えることを楽しもう

データ型って何？

これも普段聞きなれない言葉かもしれませんが、Excelを使ったことがある人は必ず触れています

端的に言うと『この値は日付です』『この値は数字です』などということです

プログラミングでは、この型がとても大事になってきます

型名	内容	例
integer	整数	1
float	浮動小数点	3.14
boolean	判定	True,False
string	文字列	ストアカ
resource	リソース	resource
array	配列	(‘1’,’2’,’3’,’4’)
object	オブジェクト	$oobj = new ClassName();
null	ヌル(大文字小文字を区別しない)	Null

例題

① $x = “2021-1-1”;

② $y = 1;

③ $z = (string)x + (int)y;

左辺の変数には何が入っていることになるでしょう？

考えることを楽しもう

入出力は意外と限定されている

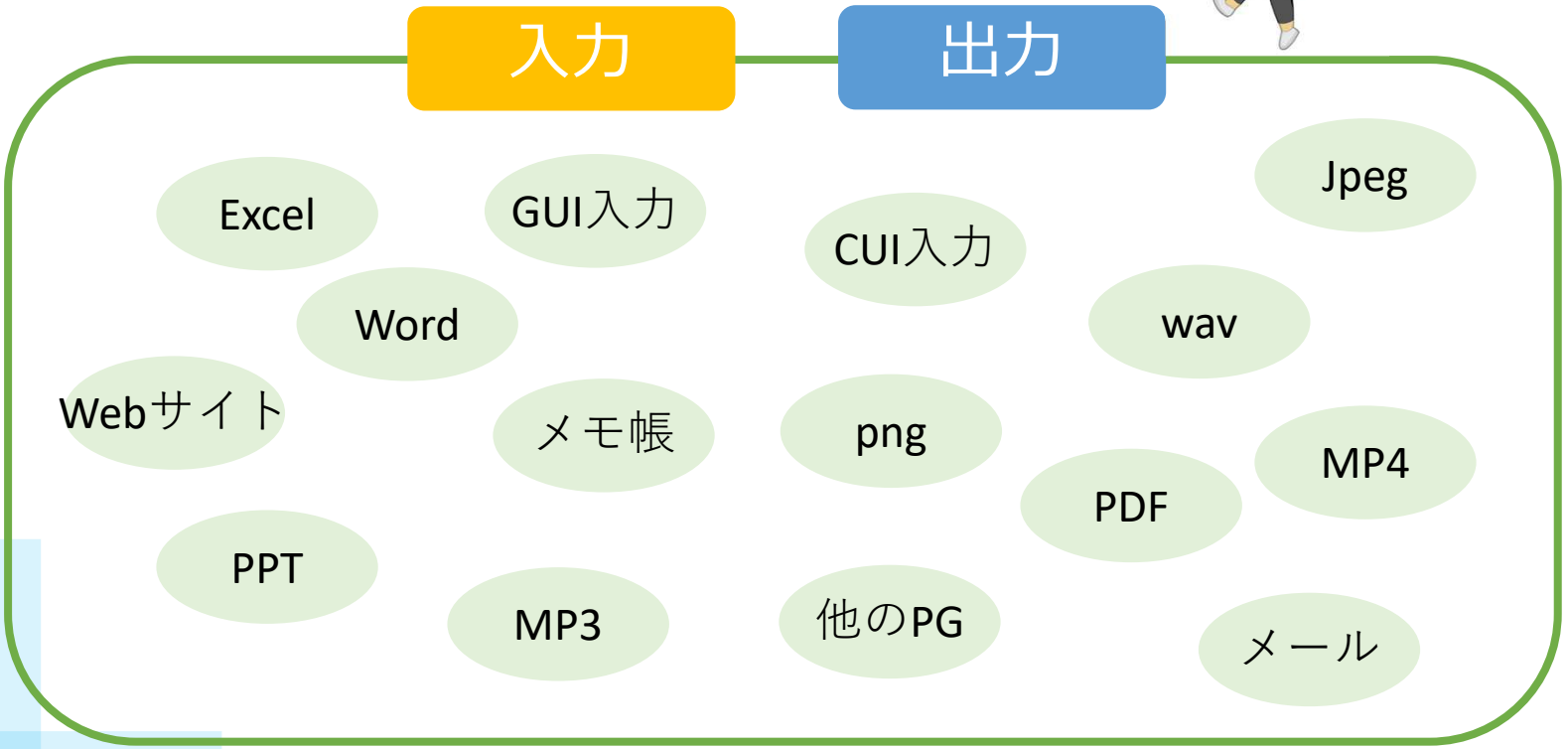
プログラムを作りたいときに大事なことは、

最初状態を設定し、最後の状態をイメージすることです

これがきっちり決まっていれば、プログラムはできたも同然です

そのぐらい大事な要素です

でもどちらの形も実はあまり種類はないです



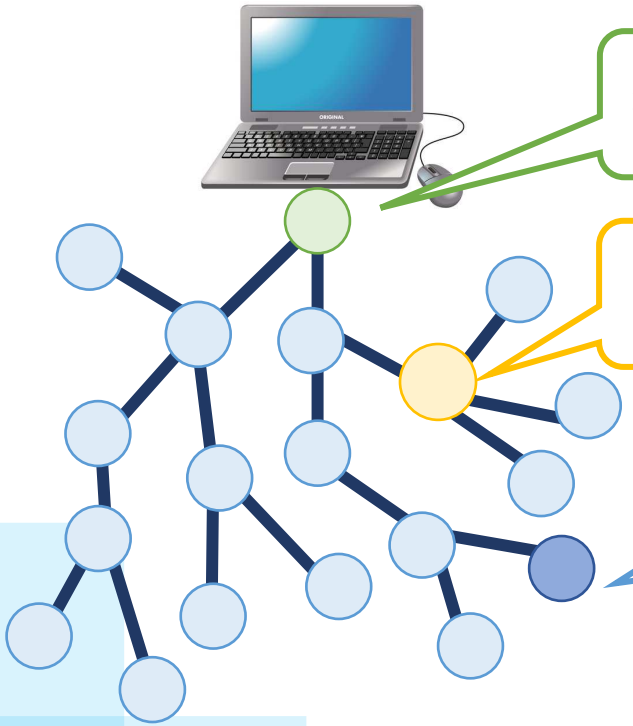
考えることを楽しもう

Pathって何?

簡単に言えば、**住所と道順の組合せのようなもの**です

パソコンのデータは必ず、ドライブといわれている記憶装置に入っており、そこから枝分枝分かれをしてDeskTopや、各フォルダにデータが置かれています

ドライブの中のディレクトリが木のように分かれていることから**ディレクトリツリー**とも言われていて、これを自在に移動することで、プログラムをより幅のある動きるようになります



カレントディレクトリから見て**メインドライブまでのパス**

絶対パス
例：C:¥Users¥XXX¥yyyy¥zzz

現在作業中のディレクトリ

カレントディレクトリ
例：.¥qqq

.は、今のディレクトリ内にある別のデータを見るという意味です
（書かなくてもいいこともある）

カレントディレクトリから見ての**何かをしたいディレクトリまでのパス**

相対パス
例：..¥aaa¥bbb¥ccc

..は、一つ上のディレクトリに移動するという意味です

考えることを楽しもう

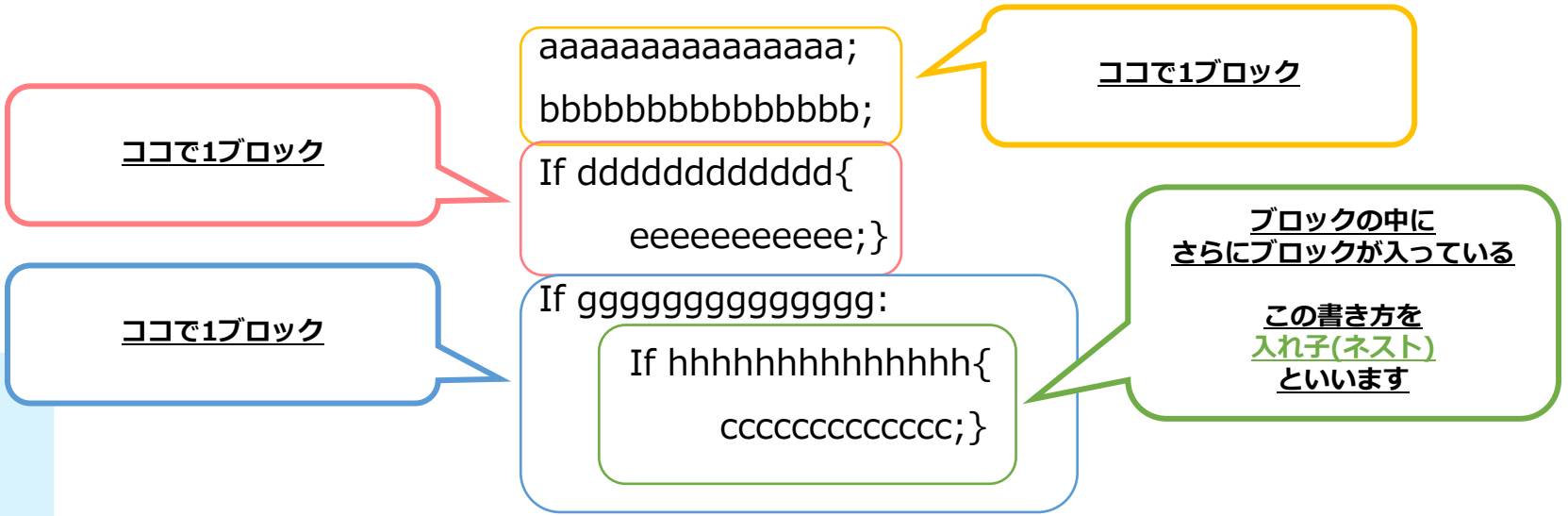
インデントの役割

プログラムを見るとよく**字下げをしている部分**がありますがあれは、決して適当に付けているわけではありません

コードをブロックのように見立てて、読みやすくしています

PHPにはインデントにプログラム的な意味はありません

ただし、**PHP以外の言語ではインデントの意味合いが変わったり、意味がなかったりするものがある**ので、注意してください



具体的なコードの書き方

書くことよりも読むことを先ずは覚えていこう

世の中には、**読み下し**という言葉があります

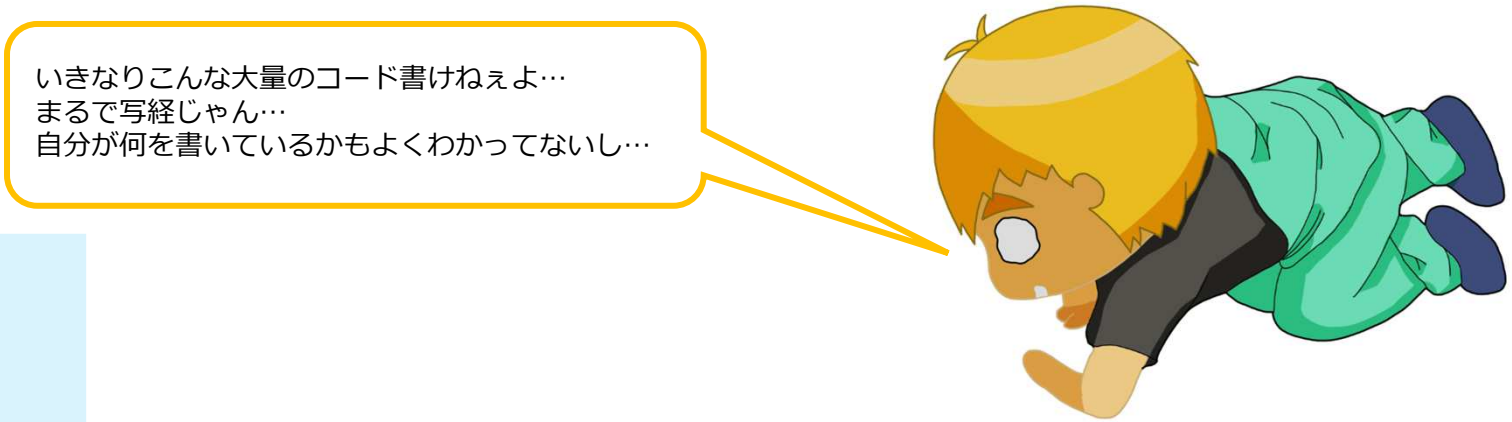
よく勉強し初めるときに一生懸命コードを書きだす方が多いですが、実はそれでは非効率です

中学生時代にやっていたテスト勉強を思い出してください

ノートに書いて覚えていたという方はいませんか？実は**PCでコードを書くのもノートに教科書の文字を写すのもやっていることはまるで同じ**です

書いて覚えるより、読んで覚える方が圧倒的に効率がいいのです

10分間で1つのプログラムコードを書くなら、10分同じコードを何度も読んで、見ないでも書けるぐらい理解したほうがいいのです



具体的なコードの書き方

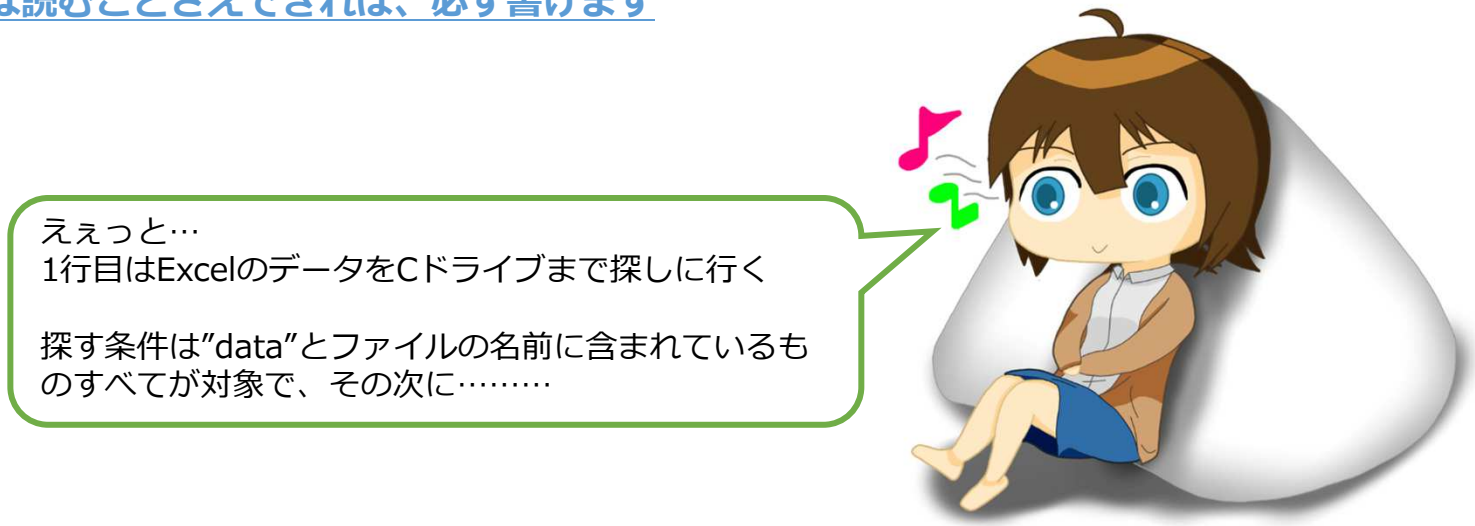
書くことよりも読むことを先ずは覚えていこう

その時の読む技術として**読み下し**があります

端的に言ってしまえば、**コードを日本語に変えて理解する**ということをします

これができれば、確実にコードが書けるようになりますし、また**似たようなプログラムと遭遇した時に何となく読める**ようになります

コードは読むことさえできれば、必ず書けます



具体的なコードの書き方

プログラムの実行の仕方

PHPはサーバ側の言語といわれていることもあり、プログラムを動かすための環境づくりをしなくてはいけません

環境作成とは、プログラムを作って動かせる状態を作ると思ってもらえれば大丈夫です

ここでは、プログラムを作成する(コードを書く) 環境をVisual Studio Code

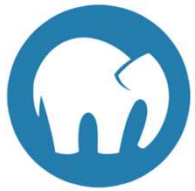
動かす環境をXAMPPもしくはMAMPとして解説していきます

作成環境

とにかくいろんな事の出来る高機能メモ帳だと思ってください

動作環境

OSによって環境が変わります



MAMP

Winows

Mac

具体的なコードの書き方

どこにどうやってかけばいいの？

プログラムを書くときにはエディターというもので書きます

極端なことを言えば、メモ帳でも書けます

ですがメモ帳だと書きづらいので、ココでは**Visual Studio Code**というもを使って書いていきます

メモ帳より高機能なメモ帳だと思ってもらえれば大丈夫です

何が高機能かというとPHPのコードを書く補助をしてくれたり、コードを識別してそれが変数なのか関数なのかなどを色分けしてくれます

PHPの入力補助の拡張機能は別のページで説明します

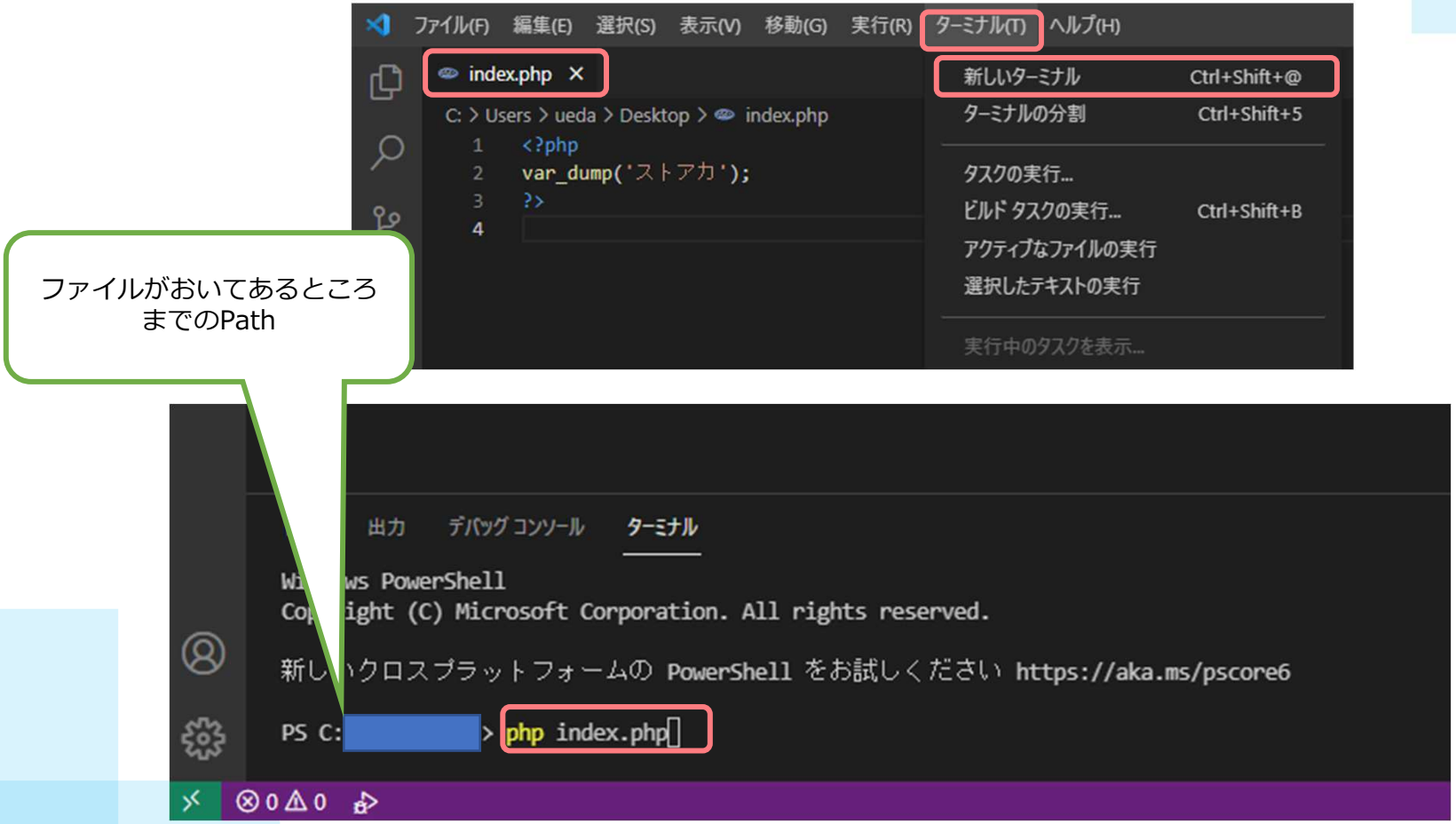
具体的なコードの書き方

どうやって動かせばいいの？

書き方はわかったけどどう動かせばいいの？

PHPはサーバー言語なので基本的にはサーバーに置かない限り動きを見れません

ですがVS-Codeのターミナルを使って、phpファイルを指定すると起動させることができます



具体的なコードの書き方

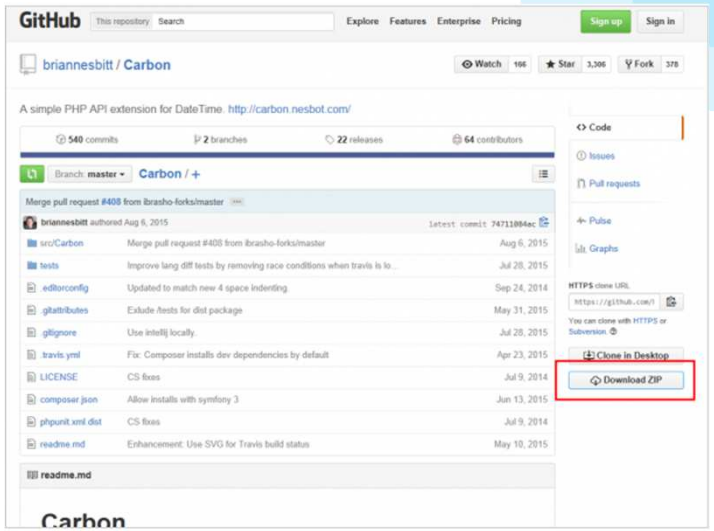
ライブラリ、モジュールの使い方

先ずは使いたいライブラリをインターネットからダウンロードして、動作環境の中に保存します

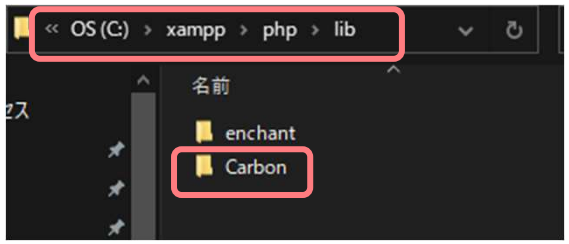
*ダウンロードしてどのフォルダを動作環境の保存するかはライブラリによって異なります

Conposerを使ったコマンドでのインストールもできます

```
$ composer require nesbot/carbon
```



必要なフォルダを動作環境内のphp内にlibというフォルダを作って入れておきます(既にフォルダがある場合はその中に)




```
<?php
require_once ../php/lib/Carbon/Carbon.php;

use Carbon¥Carbon;

echo Carbon::tomorrow();
?>
```

読み下し

カレントディレクトリで、使いたいライブラリまでのルートを指定(*どこに保存いているかで変わります)

Carbonというライブラリ中からCarbonを使います

Carbonの機能を使って明日の日付を拾い、コンソールに表示しなさい

具体的なコードの書き方

入力関係

コンソール内での入力

```
<?php
var_dump(“値を入れて”);
$msg = fgets(STDIN);
var_dump($msg);
?>
```

読み下し

コンソールに”値を入れて”と表示して、文字を入力できるようにする
入力形式は標準入力です
入力した文字はmsgに入れなさい
msgに入っている値をコンソールに表示しろ

出力関係

コンソール内での文字の表示

```
<?php
var_dump(“ストアカ”);
?>
```

読み下し

ストアカを表示しろ
型はvriant型でね

```
<?php
echo (“ストアカ”);
?>
```

読み下し

ストアカを表示しろ

具体的なコードの書き方

計算関係

四則演算の方法

右のような演算子を使って、計算をすることができるよ
慣れないものもあると思うので、使ってみて覚えていこう

演算子	使用方法	例
+	左辺に右辺を足す	5 + 2
-	左辺から右辺を引く	5 – 2
*	左辺と右辺を掛ける	5 * 2
/	左辺から右辺を割る	5 / 2
%	左辺を右辺で割った余りを計算	5 % 2
**	左辺を右辺乗する	5 **2

```
<?php
var_dump(“10 + 5”);
?>
```

読み下し

10 + 5という文字を表示しろ

```
<?php
var_dump(10 + 5);
?>
```

読み下し

10 + 5の計算結果を表示しろ

```
<?php
$sum = 10 + 5;
var_dump($sum);
?>
```

読み下し

sumに計算式の答えを入れなさい
それを表示しろ

具体的なコードの書き方

処理関係

文字を表示する

```
<?php
var_dump(“ハロー!”);
?>
```

読み下し

xに”ストアカ”を入れなさい
yに”楽しい!!”を入れなさい
xとyの文字をつなげて表示しろ

文字をつなげる

```
<?php
var_dump(“値を入れてね”);
$msg = trim(fgets(STDIN));
var_dump(“入力したのは” . $msg . “です”);
?>
```

読み下し

コンソールに”値を入れてね”と表示して、文字を入力できるようにする
入力形式は標準入力です
入力した文字はmsgに入れなさい
msgに入っている値をコンソールに表示しろ

trimは、余計な空白を取り除けという役割がある

【.】は、文字列を連結する役割がある

具体的なコードの書き方

処理関係

文字を置き替える

```
<?php
$moji = “ストアカを受けよう”;

$moji = str_replace(“ストアカ”,“講座
",$moji);
var_dump($moji);
?>
```

読み下し

mojiにストアカを受けようと入れる

mojiの中にストアカの文字列があるかを調べて、あればそれを講座という文字列入れ替えてmojiに入れる

コンソールにmojiに入っている値を表示しろ

文字の中に特定の文字がどれだけ入っているか

```
<?php
$moji = ‘ストアカ’;
if( preg_match (‘/ス/’,$moji)){
        var_dump(‘あたり’);
}

if( !preg_match (‘/ス/’,$moji)){
        var_dump(‘はずれ’);
}
?>
```

読み下し

mojiにストアカを入れなさい

もしmojiにスが含まれていたら’あたり’と表示しろ

もしmojiにスが含まれていないなら’はずれ’と表示しろ

具体的なコードの書き方

処理関係

文字列の数を数える

```
<?php
echo mb_substr_count(“abcabcaa”, “a");
?>
```

読み下し

abcabcaaの文字列の中にaが何個入っているかをコンソールに表示しろ

配列の作成

```
<?php
$week = [‘月’,’火’,’水’,’木’,’金’];
$week[1] = 炎;
array_pop($week);
array_push($week,’土’)
?>
```

読み下し

Xに5という値を入れる
もし、xの値が3より小さいのならあたりを表示しろ
違うならはずれを表示しろ

具体的なコードの書き方

処理関係

配列内の文字検索

```
<?php
$array = ['JAVA', 'PHP', 'Python'];

$result = array_search('PHP', $array);
var_dump($result);
?>
```

読み下し

arrayという配列に値を入れなさい

その中にPHPという文字列があるかを判断して
結果をresultへ入れなさい

コンソールに結果に応じてTrueかFalseを表示しろ

連想配列

連想配列の時にはダブルアロー演算子を使います

```
<?php
$array = [
        [‘name’=>'JAVA’, ‘num’=>1],
        [‘name’=>'PHP’, ‘num’=>2]
];
var_dump($array[1][‘name’]);
?>
```

読み下し

arrayという配列に値を入れなさい

配列の中に配列を入れて、データごとにキーを付けるて各王する

コンソールに配列の2番目のnameの項目を表示しろ

この配列で、このキーなら…
これなら連想しやすいから配列の中身を把握しやすいね

具体的なコードの書き方

処理関係

条件分岐

```
<?php
$x = 5;
if ( $x < 3) {
        var_dump(“あたり”);
}else{
         var_dump(“はずれ”);
}
?>
```

読み下し

Xに5という値を入れる
もし、xの値が3より小さいのならあたりを表示しろ
違うならはずれを表示しろ

演算子	使用方法	例
<	左辺の方が右辺より小さい	a <b
<=	左辺は右辺以下	a <= b
>	左辺の方が右辺より大きい	a > b
>=	左辺は右辺以上	a >= b
==	左辺と右辺は等しい	a == b
!=	左辺と右辺は等しくない	a != b
and	論理積	a and b
or	論理和	a or b
xor	排他的論理和	a xor b

読み下してみると意外と覚えやすかったりするよ

具体的なコードの書き方

処理関係

条件分岐

```
<?php
$x = 5;
if ( $x < 3) {
        var_dump(“あたり”);
}elseif ($x > 4 and $x < 6){
        var_dump(“これや!!”);
}else{
        var_dump(“はずれ”);
}
?>
```

読み下し

Xに5という値を入れる
もし、xの値が3より小さいのならあたりを表示しろ
違っていて、もし、4より大きくて、かつ、６より小さいなら、これや!!を表示しろ
違うならはずれを表示しろ

複雑に見えても一行ずつ読み下していけば大丈夫やよ

具体的なコードの書き方

処理関係

複数分岐

```
<?php
$x = 5;

switch ($x){
case 5:
        var_dump(“あたり”);
Break;

Case $x > 5:
       var_dump(“半分あたり”);
break;

default:
       var_dump(“はずれ”);
break;
}
?>
```

読み下し

Xに5という値を入れる

xに5が入っていれば、あたりと表示しろ
xに5以上の値が入っていたら半分あたりと表示しろ
どれでもないならはずれと表示しろ

プログラミング言語によっては、このロジックはなかったりするよ

具体的なコードの書き方

処理関係

++という書き方をインクリメントというよ
--はデクリメントで、どちらも現状から1ずつ変化させなさいという意味やよ

配列を使って繰り返し

```
<?php
$week = [‘月’,’火’,’水’,’木’,’金’];
$week[1] = 炎;

array_pop($week);
array_push($week,’土’);
array_reverse($week);
array_shift($week);
array_marge($week,[‘日’,’天’]);

foreach ($week as $day){
    var_dump($day . 'です');
}
?>
```

読み下し

weekという配列に値を入れる
weekの配列の先頭から2番目の値に炎を入れる

weekの配列の末尾の値を取り出す
weekの配列の末尾に土という値を入れる
weekの配列内の値の並びを逆にする
weekの配列の先頭の値を取り出す
weekの配列の後ろに新しい配列を加える

weekという配列の値を一つずつdayという変数に入れる
コンソール内にdayに入っている値と文字列を組み合わせて表示しろ
weekという配列内の値がなくなるまで繰り返す

繰り返し(上限あり)

```
<?php
for ($i = 0; $i <5; $i++){
    var_dump($i . '回目です');
}
?>
```

読み下し

変数iに先ずは0を入れなさい
iを1ずつ増やしていき、iが5未満の間は間の処理を繰り返しなさい

コンソールに変数iと文字の組合せを表示しろ

具体的なコードの書き方

処理関係

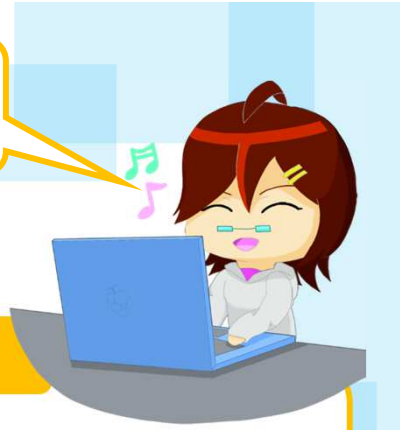
条件を使って繰り返し

```
<?php
$money = 3000;

while ( $money > = 0 ){
        var_dump('会計完了');
        $money -= 500
}
?>
```

+=や、*=といった書き方を
複合演算子と言います

$money = $money + 500と同じ意味で
簡単に書けるね



読み下し

moneyという変数に値を入れなさい

moneyの値が0になるまで次のことを繰り返しなさい
コンソールに会計完了という値を入れなさい
moneyから500を引いてmoneyに入れなさい
ここまでの作業を繰り返しなさい

繰り返し条件から抜ける、処理を飛ばす

```
<?php
$week = ['月','火','水','木','金'];

foreach ($week as $day){
        if ($day != '水'){
                var_dump($day . 'です');
        }else{
                break;
        }
}
?>
```

読み下し

weekという配列に値を入れる

weekという配列の値を一つずつdayという変数に入れる
もし、dayの中の値が水以外であれば、次のことを実行しないさい
コンソール内にdayに入っている値と文字列を組み合わせて表示しろ
違うなら繰り返しから抜けないさい

weekという配列内の値がなくなるまで繰り返す

具体的なコードの書き方

関数関係

　プログラムを作っていくうちに何度も同じコードを書くことが出てきたり、それがある程度まとまった形で何度も出てくることがあります

　そんなときにそのコードのまとまりに名前を付けて、何度も使いまわせるようにしたものを関数といいます

関数の作り方

```
<?php
function sutoaka () {
    var_dump('ストアカ');
    var_dump('受けたい');
}
sutoaka();
?>
```

読み下し

sutoakaという関数を作ります
コンソールにストアカと表示しろ
コンソールに受けたいと表示しろ

sutoakaという関数を実行しなさい

何度も呼び出せるんやね
何度も同じコード書かなくてもええね

具体的なコードの書き方

関数関係

関数の作り方(引数あり)

```
<?php
function sutoaka ($moji) {
    var_dump($moji . 'ストアカ');
    var_dump('受けたい');
}
sutoaka(‘私も’);
?>
```

読み下し

sutoakaという関数を作ります
関数を呼び出すときに引数としてmojiを使います

コンソールにmojiの値とストアカを組み合わせて表示しろ
コンソールに受けたいと表示しろ

sutoakaという関数を実行しなさい
実行するときに’私も’を渡しなさい

関数の作り方(引数複数)

```
<?php
function sutoaka ($moji,$text) {
        $msg = <<< EOM
        {$moji}ストアカ受けたい{$text}¥n
        EOM;
        echo $msg;
}
sutoaka(‘私も’,’かも？’);
?>
```

読み下し

sutoakaという関数を作ります
関数を呼び出すときに引数としてmojiとtextを使います

長文文字列を作ります
引数を使って文字列を作ります
作った文字列をコンソールに表示しろ

sutoakaという関数を実行しなさい
実行するときに’私も’と’かも?’を渡しなさい

具体的なコードの書き方

関数関係

関数の作り方(引数、戻値あり)

```
<?php

function sutoaka ($moji) {
    $text = $moji . ‘ストアカ受けたい';
    return $text;
}
echo sutoaka(‘私も’);

?>
```

読み下し

sutoakaという関数を作ります
関数を呼び出すときに引数としてmojiを使います

mojiの値とストアカを受けたいを組み合わせて、textという変数に入れないさい
呼び出し元にtextという変数を渡しなさい

sutoakaという関数を実行して、戻ってきた結果をコンソールに表示しなさい
実行するときに’私も’を渡しなさい

具体的なコードの書き方

高度のな処理

他のモジュールを呼び出して実行

```
<?php
function sutoaka ($moji) {
    $text = $moji . 'ストアカ受けたい';
    return $text;
}
?>
```

読み下し

sutoakaという関数を作ります
関数を呼び出すときに引数としてmojiを使います

mojiの値とストアカを受けたいを組み合わせて、textという変数に入れないさい
呼び出し元にtextという変数を渡しなさい

```
<?php
require_once __DIR__.'/教材用.php';

echo sutoaka('私も');
?>
```

読み下し

一度だけ教材用.phpのモジュールを呼び出す
モジュールのパスは、今のフォルダと同じディレクトリにあります

sutoakaという関数を実行して、戻ってきた結果をコンソールに表示しろ
実行するときに'私も'を渡しなさい

あなたの関数使わせてもらうね

わかった
引数くれたら処理して結果を渡すね

具体的なコードの書き方

HTMLとの連携

PHP最大の強みはHTMLのコードの中に埋め込めるという部分です

これまで、コンソール内でしか表示できなかったものも、HTMLのコード内に埋め込むことによってブラウザ上で処理内容を表示することができます(ファイルの拡張子は、.htmlから.phpに変えてください)

ページの表示の方法は別のページに詳しく記載します

WEBサーバを起動

MAMPやXAMPPを起動させて、PCのフォルダ内を仮想のローカルサーバとして使えるようにする

*詳しくは別のページで説明します

MAMP

具体的なコードの書き方

HTMLとの連携

値を表示する

```
<!DOCTYPE html>

<html lang="jp">

<head>
        <meta charset="UTF-8">
        <title>Document</title>
</head>

<body>
        <?php
        echo 'ストアカ'
         ?>
</body>

</html>
```

読み下し

HTMLとしてのフォーマットで記載する

ヘッダー情報を記載する

ボディーの情報を記載する

ココからがphpのコードです
ストアカという文字列を表示しろ
ココまでがphpのコードです

ここまでがHTMLのコードです

今まで初めて終わりに書いてあった
<?phpと?>はその間に書かれている
ものがphpのプログラムであるという
目印だったんだね

具体的なコードの書き方

HTMLとの連携

Getﾒｿｯﾄﾞ

フォームを利用してプログラムを動かす

```
<!DOCTYPE html>
<html lang="jp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <h1>ストアカ</h1>
    <form action="教材用.php"
method="get">
    <input type="text" name="comment">
    <input type="submit" value="送信">
    </form>
</body>
</html>
```

読み下し

HTMLとしてのフォーマットで記載する

ヘッダー情報を記載する

ボディーの情報を記載する

見出しはストアカと表示しろ

フォームを作ります
アクションがあったら教材用.phpのモジュールを使います

データの渡し方の方法はgetというメソッドを使います

テキストを入力できる部分を作ります
nameはcommentとします

ボタンを作り、送信という文字を表示しろ
ここまでがフォームです

ここまでがボディです

ここまでがHTMLのコードです

```
<?php
$comment = $_GET['comment'];
echo $comment;
?>
```

読み下し

Getメソッドでcommentというnameに入っている値をもらって、commentという変数に入れます
画面にcommentに入っている値を表示しろ

具体的なコードの書き方

HTMLとの連携

POSTﾒｿｯﾄﾞ

フォームを利用してプログラムを動かす

```
<!DOCTYPE html>
<html lang="jp">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <h1>ストアカ</h1>
    <form action="教材用.php"
method=“post">
    <input type="text" name="comment">
    <input type="submit" value="送信">
    </form>
</body>
</html>
```

読み下し

HTMLとしてのフォーマットで記載する

ヘッダー情報を記載する

ボディーの情報を記載する

見出しはストアカと表示しろ

フォームを作ります
アクションがあったら教材用.phpのモジュールを使います

データの渡し方の方法はpostというメソッドを使います

テキストを入力できる部分を作ります
nameはcommentとします

ボタンを作り、送信という文字を表示しろ
ここまでがフォームです

ここまでがボディです

ここまでがHTMLのコードです

```
<?php
$comment = $_POST['comment'];
echo $comment;
?>
```

読み下し

POSTメソッドでcommentというnameに入っている値をもらって、commentという変数に入れます
画面にcommentに入っている値を表示しろ

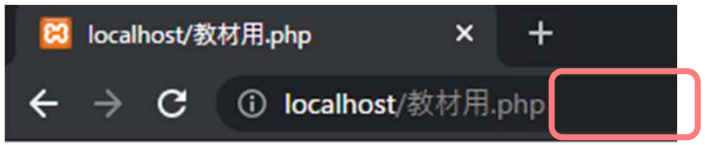
具体的なコードの書き方

HTMLとの連携

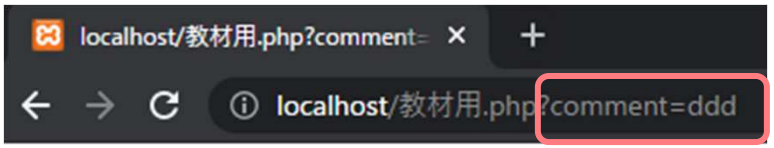
GETメソッドとPOSTメソッドの違いは何？

プログラムを起動させて気づいたかもしれませんが**ページが表示されているときのURLが違います**

POSTメソッド



GETメソッド



動的なページの場合は、入力した値によって表示が変わります

個人がどんなページを検索したかを秘匿したいときにはPOSTメソッドを使い、逆に多くの人の動的なページの中で**検索する条件とともにページを知ってもらいたいときにはGETメソッド**を使うなど、目的に応じてメソッドを変えると便利です

PHPには**スーパーグローバル変数**というものがあって、全て連想配列になっています

今回のgetﾒｿｯﾄﾞやpostﾒｿｯﾄﾞはまさに**そのスーパーグローバル変数**を使ってるよ(全9種類)

$GLOBALS、$_SERVER、$_GET、$_POST
$_FILES、$_COOKIE、$_SESSION、
$_ENV
$_REQUEST

具体的なコードの書き方

HTMLとの連携

セキュリティのための一工夫

```
<?php
function h($str){
    return
htmlspecialchars($str,ENT_QUOTES,'UTF-8');
}
$comment = h ($_POST['comment']);
echo $comment
?>
```

読み下し

hという関数を作る
読み出し元に渡された文字列をUTF-8にエンコードして戻す

POSTメソッドでcommentというnameに入っている値をもらって、hという関数で加工してからcommentという変数に入れます
画面にcommentに入っている値を表示しろ

ストアカ

<script>alert('警告');</script> 送信

左のように入力欄に変なプログラミングコードが入れられてもこれなら動作しないで文字列として認識してくれるからWEB攻撃に耐えられるね

具体的なコードの書き方

PHPならではのルール

constとdefineの違い

```
<?php
const TEST1 = 'ストアカ';
echo TEST1;

define('TEST2', '受講');
echo TEST2;
?>
```

読み下し

定数としてストアカを入れます
コンソールに定数の値を表示しろ

関数を使って定数として受講を入れます
コンソールに定数の値を表示しろ

const変数は、定数としての扱いで変更することはできないよ
だから$はつかないんだ
わかりやすいように全て大文字で書くよ

define関数も、一度値を入れると変更することはできないよ

具体的なコードの書き方

PHPならではのルール

Classの使い方(オブジェクト指向)

2ページに渡って説明してます
番号順にコードが起動していきます

```php
<?php
class Dog {

        private $name;

        public function __construct($name) {
                $this->name = $name;
    }
        public function getName() {
                return $this->name;
    }
}

class Dachshund extends Dog {

        private $sound = "Bow!!";

        public function barking() {
                return $this->sound;
    }
}
?>
```

1

3

読み下し

DogのClassを定義する

$nameというプロパティ(特徴)を定義する

インスタンス化するときに引数を受け取れるようにする
プロパティに入っている値をnameに渡す

名前を取得するメソッドを定義する
メソッドに入っている値をnameに返す

Dachshund Classを作って、Dog Classを継承する

$soundという新しいプロパティ(特徴)を定義する
値として”Bow!!”を入れる

barkingという新たなメソッドを定義する
メソッドに入っている値をsoundに返す

$thisはインスタンス自身のメソッドやプロパティに入っている値を表してるよ

self::も同じ感じだけど、こっちはインスタンス化されていないものを呼び出すよ

具体的なコードの書き方

PHPならではのルール

Classの使い方(オブジェクト指向)

```
<?php
$dog = new Dog("John");

$dog_name = $dog->getName();

echo "名前: " . $dog_name . "¥n";


$dachshund = new Dachshund("Doh");

$dachshund_name = $dachshund-
>getName();

$dachshund_sound = $dachshund-
>barking();

echo "名前: " . $dachshund_name . "¥n 鳴き
声: " . $dachshund_sound;
?>
```

2

4

読み下し

Dog Classをインスタンス化する
Johnという値を渡す

インスタンス化したメソッドを使って名前を取得する
$dogに戻り値を渡す

コンソールに変数の値と文字列を組み合わせたものを表示しろ

Dachshundをインスタンス化する
Dohという値を渡す

インスタンス化した後、Dog Classから継承したgetNameメソッドで名前を取得する
$dachshund_nameに戻り値を返す

インスタンス化したbarkingメソッドを使って鳴き声を取得する
$dachshund_soundに戻り値を返す

コンソールに変数の値と文字列を組み合わせたものを表示しろ

具体的なコードの書き方

PHPならではのルール

アロー演算子とコンストラクタ

```
<?php
class Person {

        private $name;
        function __construct($name) {
                $this->name = $name;
    }
        function getName() {
                return $this->name;
    }
        function introduceSelf() {
                echo "私の名前は". $this->name ."で
す".PHP_EOL;
    }
}

$taro = new Person("太郎");
$test_name = $taro->getName();
echo $test_name.PHP_EOL;

$taro->introduceSelf();
?>
```

__construct は、オブジェクトの初期化という意味で必ず最初に書かないといけないもの!!

-> は、左辺にインスタンスの値を置いて右辺で呼び出す!!

読み下し

Personというクラスを作る
nameという変数を作る
コンストラクタでインスタンス生成時に名前を設定できるようにしろ

名前を取得するメソッドを定義する
メソッドに入っている値をnameに返す

自己紹介のメソッドを定義する
コンソールに変数と文字れるの組合せを入れなさい
最後に改行を入れなさい

Person Classをインスタンス化する
太郎という値を渡す
インスタンス化したメソッドを使って名前を取得する
$test_nameに戻り値を渡す
コンソールに変数の値を表示しろ

インスタンス化したメソッドを使う

具体的なコードの書き方

PHPならではのルール

PublicとprotectedとprivateのPHPの使い方の違い

```
<?php
class Kouza{

  private $text;

  public function __construct($t){
    $this->text = $t;
  }

  public function get_info(){
    return $this -> text ;
  }

}

$test = new Kouza('ストア力');
echo $test->get_info();
?>
```

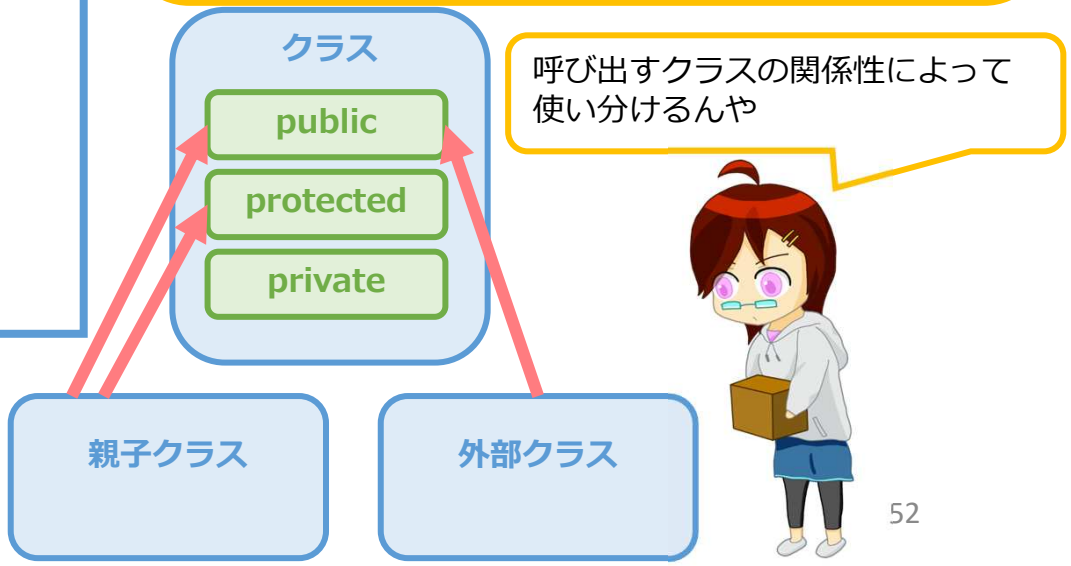
読み下し

Kouzaクラスを作ります

textという変数を作る(このクラス内でしか使えない)
コンストラクタでインスタンス生成時に値を設定できるようにしろ(どこからでも呼び出せる)
メソッドに入っている値を$tに返す

Get_infoメソッドを定義する(どこからでも呼び出せる)
クラスの値を渡す

Person Classをインスタンス化する
太郎という値を渡す



具体的なコードの書き方

PHPならではのルール

クラスをインスタンス化しなくても使えるstatic

```
<?php
class Text {
  public static function getMessage() {
    echo “ストアカ";
  }
}

Text::getMessage();
?>
```

読み下し

Textクラスを作ります

getMessageメソッドを定義する
(どこからでも呼び出せ、かつインスタンス化しなくてもいい)
コンソールにストアカと表示しろ

Textクラス内のメソッドを呼び出す

本日のまとめと感想

本日のまとめ

・言語の特徴をとらえて、どんなことができるかを知りましょう

・プログラムを組むときには、先ずは言語化をしていきましょう

・コードは書くよりも、何度も読み下しをした方が身に付きます

・コードを【オブジェクト】に置き換えてとらえるとワンステップアップできます

最後に皆さんの今日のスキルを

いつ、どんな場面で使おうと思ったかを教えてください

Field-UP

公式LINEページ

教室ページ

https://www.street-academy.com/steachers/465170

上田　敬介 (うえだ　けいすけ)

岐阜県岐阜市出身の少しマイペースな人物

技術や知識は単独で、覚えるより、絡めて覚えたほうが効率がいいし、
身に付くということを一緒に体感していきましょう

経験だけでなく、国家資格、民間資格を取得している技術のため、品質には自信があります

■健康予防管理専門士
■エネルギー管理士
■ITパスポート
■米国NLPプラクティショナー(バンドラーライン)
■アロマセラピスト 1級
(全55資格取得済み)

各種インストール方法

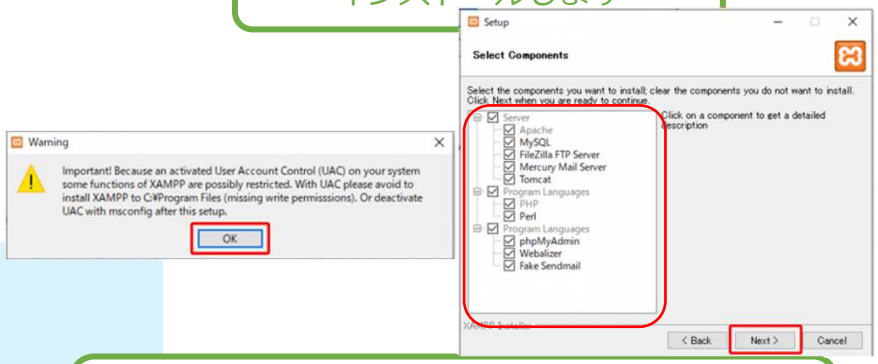
XAML、MAMPのインストール方法

Windowsは、XAMPP。Macは、MAMPとOS別に環境作成をオススメします

これがあるとサーバの代わりになって、プログラムを実行などをしてくれます

https://www.apachefriends.org/jp/index.html

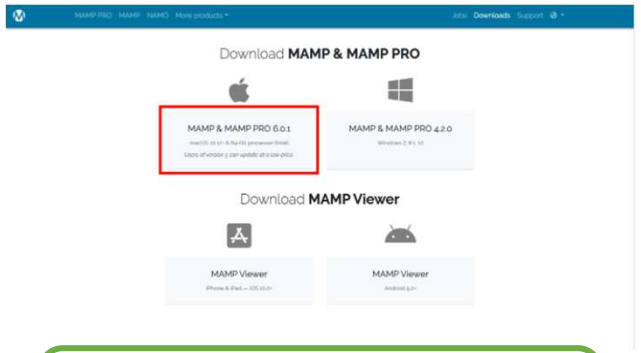
ダウンロードできたらexeで
インストールします



Cドライブ直下にインストールするためOKです
インストーラーがでたらすべての項目にチェック
を入れてインストール

順番にやっていこう

https://www.mamp.info/en/mac/

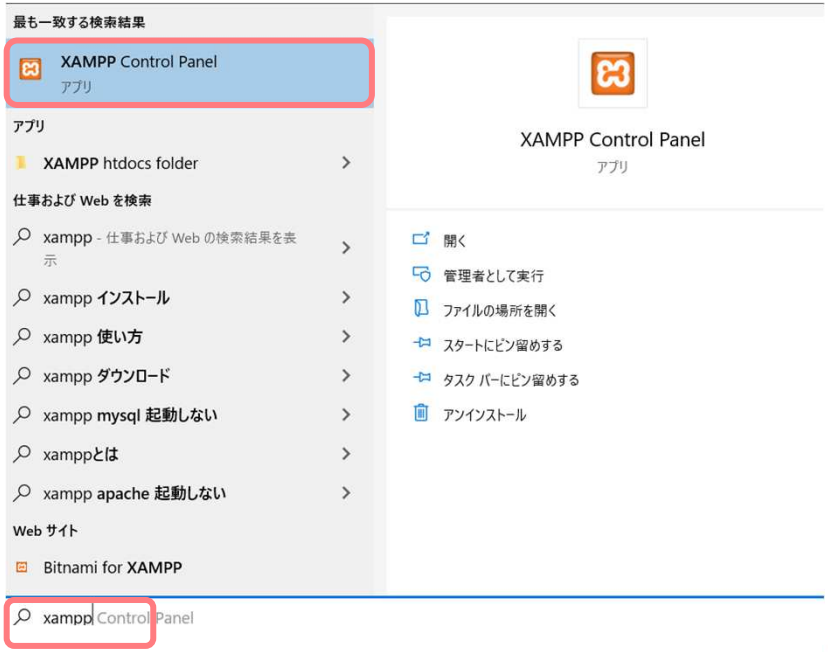


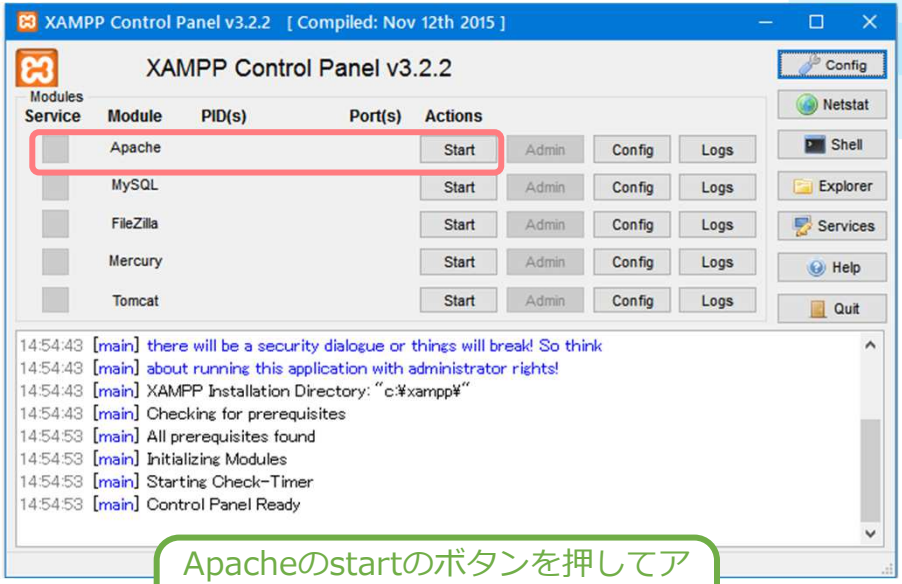
ダウンロードできたらexeで
Cドライブ直下インストールします

各種インストール方法

XAMPPの起動方法

メニュー画面からXAMPPを起動させてから各アプリケーションを起動させます





Apacheのstartのボタンを押してアプリが立ち上がればWEBサーバとして利用できます

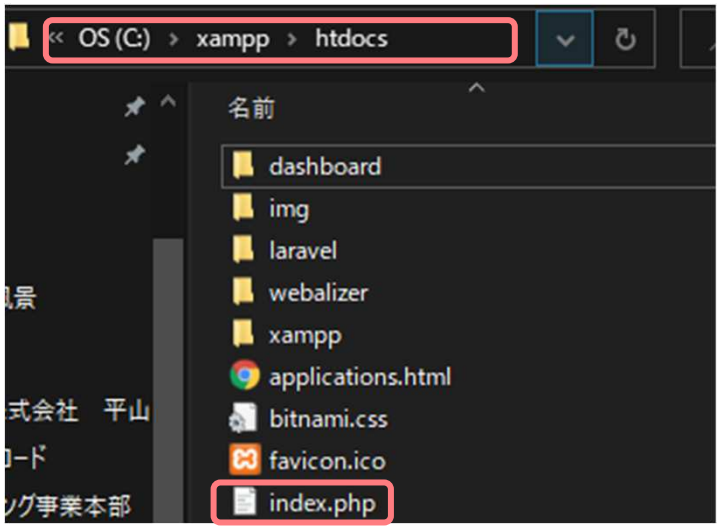
Apacheとは、WEBサーバアプリケーションのことやよ
サーバも普通のパソコンと同じでアプリケーションでいろいろ動いとるんやね

各種インストール方法

XAMPPの起動方法

サーバが立ち上がったらいよいよプログラムの実行です

XAMPPのフォルダ内の【htdocs】内に作ったプログラムやフォルダをおけば準備完了です



ブラウザを立ち上げて、URLに【http://localhost/dashboard/】と入力するとXAMMP環境が出来上がってることが確認できます

【http://localhost/XXXXXXXX/】のXXXXXXの部分をフォルダやファイル名にするとコードが実行されてブラウザに表示されます

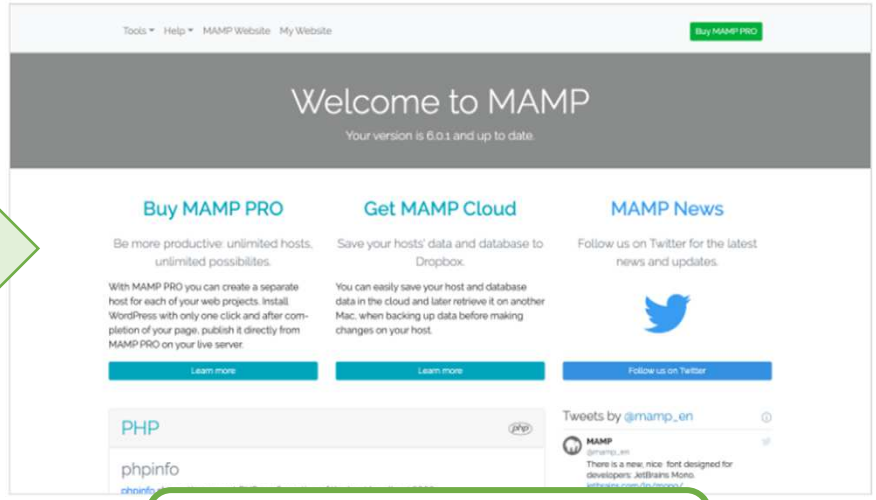
各種インストール方法

MAMPの起動方法

インストールしたMAMPを起動させて【Start Servers】を押します

ApacheとMyAQLにランプがついていればサーバーは起動してます

MAMPのフォルダ内の【htdocs】内に作ったプログラムやフォルダをおけば準備完了です



起動するとこんな画面が出ます

立ち上がったブラウザのURL【http://localhost:8888/MAMP/?language=English】のMAMPから後半をフォルダに格納したファイルやフォルダ名にするとコードが実行されます

各種インストール方法

Composerのインストール方法

サイトでOS別インストール方法でインストールしてください

https://getcomposer.org/doc/00-intro.md

Installation - Windows

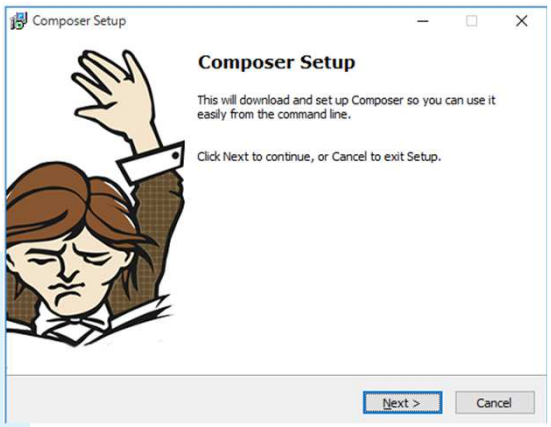
Using the Installer

This is the easiest way to get Composer set up on your machine.

Download and run Composer-Setup.exe. It will install the latest Composer version and set up your PATH so that you can call `composer` from any directory in your command line.

Note: Close your current terminal. Test usage with a new terminal: This is important since the PATH only gets loaded when the terminal starts.

ダウンロードできたらexeで
Cドライブ直下インストールします



順番にやっていこう

Installation - Linux / Unix / macOS

Downloading the Composer Executable

Composer offers a convenient installer that you can execute directly from the command line. Feel free to download this file or review it on GitHub if you wish to know more about the inner workings of the installer. The source is plain PHP.

There are in short, two ways to install Composer. Locally as part of your project, or globally as a system wide executable.

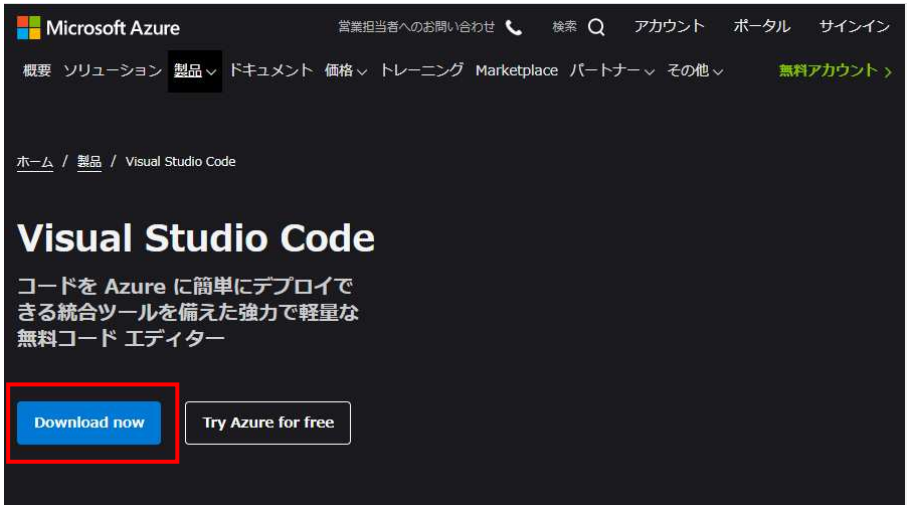
```
php composer-setup.php --install-dir=bin --filename=composer
```

ターミナルでコードを実行

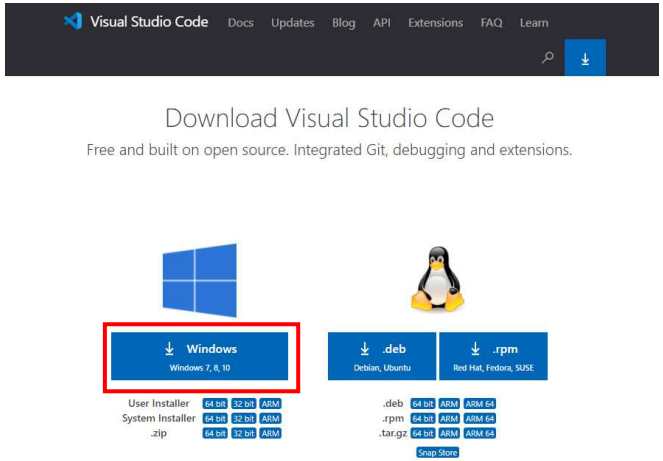
各種インストール方法

Visual Studio Code

Visual Studio CodeのWebサイトにアクセスして、【Download now】をクリックします



【Windows】のダウンロードマークをクリックし、Visual Studio Codeのインストーラーのダウンロードを開始します



各種インストール方法

Visual Studio Code

Visual Studio Codeのインストーラーをダブルクリックし、インストーラーを起動します

【同意する】を選択し、【次へ】をクリックします

各種インストール方法

Visual Studio Code

インストール先の任意のディレクトリを指定し、【'次へ】をクリックします

スタートメニューフォルダーを作成する任意のディレクトリを指定し、【次へ】をクリックする

【PATHへの追加】にチェックを入れ、【次へ】をクリックする

【インストール】をクリックすると、インストールが開始される

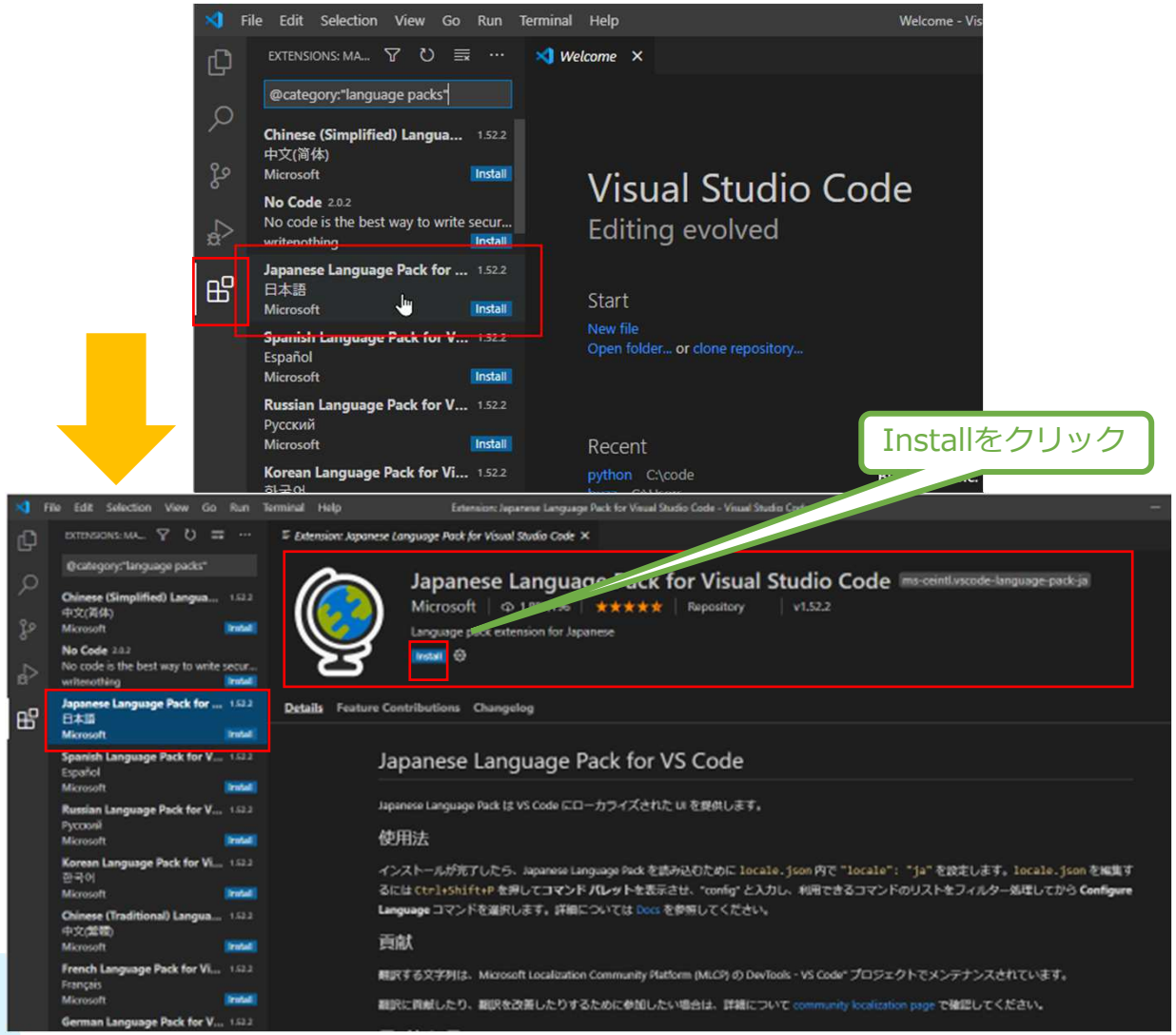
【Visual Studio Code セットアップウィザードの完了】が表示されますので、【完了】をクリックする

これでVisual Studio Codeのインストールは完了です

各種インストール方法

Visual Studio Code　日本語拡張機能

VS-Codeの日本語設定は拡張機能でできます



各種インストール方法

Visual Studio Code　PHP拡張機能

VS-Codeの拡張機能でPHPをよりコーディングングしやすくできます