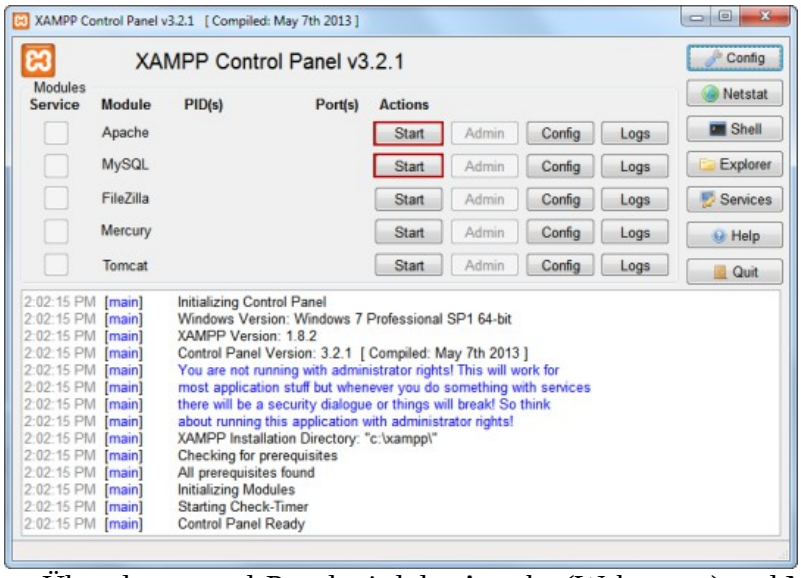
PHP Einführung

1.) Was brauche ich?

a) Editor (z.B. notpad++)
b) Webbrowser (Chrome, Firfox)
c) Webserver (XAMPP) ist auf dem Schulrechner vorinstalliert



Über das control-Panel wird der Apache (Webserver) und MySQL (Datenbanksystem) gestartet.

Dateien kommen in das Verzeichnis: „C:\XAMPP\htdocs“
hier können auch Unterverzeichnisse verwendet werden.

Die Seiten werden im Webbrowser aufgerufen:

Zum Beispiel:
C:\XAMPP\htdocs\index.html → 127.0.0.1/index.html
C:\XAMPP\htdocs\meineSuperSeite\start.html → 127.0.0.1/meineSuperSeite/start.html

2.) Variablen

Variablen in PHP sind mit dem $-Zeichen gekennzeichnet.

Beispiel:

```
$name = “Herr Anske“;          // Zeichenketten in Anführungszeichen

echo “Hallo $name“;            // Variable kann im echo-Befehl ausgegeben werden.

echo “Hallo “.$name;           // mit einem Punkte können 2 Zeichenketten aneinander
                               // gehängt werden.

$summe = 15;                   //Eine Integer Variable
$antwort = "Ganz viel Text";   //Ein String
$ergebnis = 15.5;              //Eine Zahl mit einem Komma (Typ: float)
$b = true;                     //wahr oder falsch → boolean
```

```
$zahl1 = 5;                          // Zahlen ohne Anführungszeichen
$zahl2 = 2;

echo $zahl1." + ".$zahl2." = ";
echo $zahl1+$zahl2;                  // gibt 5 + 2 = 7 aus.

$nummer = 7;                         //in der Variable wird der Wert 7 gespeichert
$nummer++;                           // ++ erhöht den Wert um 1 auf 8
$zahl = $zahl * $zahl++;             //→8*9 =72 →72 wird in der Variable Zahl gespeichert
```

Erweitere das Programm, so dass die 7er Malfolge ausgeben wird. Tipp: Schau dir Schleifen an.

3. Verzweigung: If / Else

Erkläre, was im Code passiert. Gehe dabei auf die Verzweigung (if/else) ein.

```
<?php
        $user = "Klaus";

        if($user == "Tom")          // der Inhalt der Variable wird mit "Tom" verglichen
         {
                 echo "Hallo Tom";
         }
        else
        {
                echo "Du bist nicht Tom!";
        }
?>
```

Operator	Name	Erläuterung
$a == $b	Gleich	Dieser Vergleich ist erfüllt, falls $a und $b den selben Wert beinhaltet. Sind die Typen der Variablen verschieden, so werden die konvertiert.
$a === $b	Identisch	Dieser Vergleich ist erfüllt, falls $a und $b den selben Typ und den Inhalt besitzen. Wäre ein Wert vom Typ int und der andere from Typ String, so würde *false* zurück gegeben werden.
$a != $b	Ungleich	Dieser Vergleich ist erfüllt, falls $a und $b nicht den selben Wert beinhaltet. Sind die Typen der Variablen verschieden, so werden die konvertiert.
$a !== $b	Identisch	Dieser Vergleich ist erfüllt, falls $a und $b einen unterschiedlichen Typ haben oder einen unterschiedlichen Wert.
$a < $b	Kleiner	$a muss kleiner als $b sein.
$a <= $b	Kleiner Gleich	$a muss kleiner oder gleich $b sein.
$a > $b	Größer	$a muss größer als $b sein.
$a >= $b	Größer Gleich	$a muss größer oder gleich $b sein.

Aufgabe:
Schreibe ein PHP-Programm, bei dem der Nutzer eine Zahl raten muss. Das Programm soll jeweils ausgeben, ob die richtige Zahl geraten wurde oder ob die zu findende Zahl größer oder kleiner ist.

Die Nutzereingabe kann mittels GET-Funktion gemacht werden (ist zwar umständlich, funktioniert aber):

$eingabe = $_GET["zahl"]; //die Variable $eingabe soll dann weiter geprüft werden.

Die URL muss so aussehen:
http://127.0.0.1/raten.php?zahl=10 //raten.php heißt die Datei, 10 ist die Eingabe.

Die zu ratende Zahl kann fest gewählt werden, so dass dein Nachbar die Zahl raten kann. Alternativ kann mit rand eine Zufallszahl bestimmt werden (Achtung: bei erneuter Eingabe und einem Refresh wird auch eine neue Zufallszahl bestimmt).

rand(1,20); //Zufallszahl zwischen 1 und 20

$_GET und $_POST

Bei der **GET-Methode** spricht man von Variablenwerten, die mittels der URL übergeben werden. Vielleicht ist euch im Browser bereits aufgefallen, dass viele URLs ein ? hinter dem Dateinamen haben gefolgt von entsprechenden Werten. Dies sind die GET-Variablen der Website. Im PHP-Script könnt ihr auf diese wie folgt zugreifen.

```
//        beispiel.php?vorname=Tim&nachname=Müller
<?php
$vorname = $_GET['vorname'];
$nachname = $_GET['nachname'];
echo "Hallo $vorname $nachname";
?>
```

Ausgabe: Hallo Tim Müller

Datenübergabe mittels $_POST
Im Gegensatz zu $_GET werden $_POST-Variablen nicht per URL, sondern per Formular übertragen (mehr Infos zu HTML-Formulare). Erstellt dazu ein HTML-Formular (seite1.php) mit dem folgenden Inhalt:

```
//seite1.php
<form action="seite2.php" method="post">
Vorname: <input type="text" name="vorname" /><br />
Namename: <input type="text" name="nachname" /><br />
<input type="Submit" value="Absenden" /> </form>
```

```
//seite2.php
<?php
$vorname = $_POST["vorname"];
$nachname = $_POST["nachname"];
echo "Hallo $vorname $nachname";
?>
```

Schleifen

Die Syntax einer while-Schleife ist wie folgt aufgebaut:

```
<?php
     while(Bedingung) {
          Anweisungen
     }
?>
```

Ein Beispiel wäre:

```
<?php
     $i = 0;
     while($i < 10) {
          echo "$i, ";
          $i++; }
?>
```

Im obigen Beispiel wird zuerst eine Zählvariable definiert, in diesem Fall die Variable $i und diese wird mit dem Wert 0 initialisiert.

Die Bedingung der while-Schleife ist $i < 10 , d.h. wir überprüfen ob in der Variable ein Wert kleiner 10 steht. Alles zwischen den geschweiften Klammern sind die Anweisungen der Schleife. Diese werden so oft wiederholt, wie die Bedingung erfüllt ist. Im obigen Beispiel wird die Variable $i ausgegeben und anschließend wird die Zählvariable um den Wert 1 erhöht. Diese Erhöhung des Wertes ist wichtig, damit die Bedingung irgendwann nicht mehr erfüllt ist. Falls Bedingung immer erfüllt ist, beispielsweise weil ihr vergessen habt die Zählvariable $i zu erhöhen, so resultiert euer PHP-Script in eine Endlosschleife und führt zu keinem Ergebnis mehr. Zum Glück bricht aber PHP nach einer gewissen Laufzeit euren Script ab.

Die Bedingung sowie die Anweisungen innerhalb der Schleife könnt ihr übrigens beliebig kompliziert gestalten. So könnt ihr beispielsweise auch mehr als nur eine Zählvariable definieren:

```
<?php
     $zaehler1 = 0;
     $zaehler2 = 0;
     $min = -20;
     $max = 30;

     while($zaehler1 < $max AND $zaehler2 > $min) {
          echo "Zaehler1: $zaehler1 ; Zaehler2: $zaehler2 <br>";
          $zaehler1 = $zaehler1+2;
          $zaehler2 = $zaehler2-3;
     }
?>
```

Ähnlich wie bei der while-Schleife, lässt sich die **for-Schleife** nutzen um Anweisungen solange auszuführen, wie eine bestimmte Bedingung erfüllt ist. Die Syntax der for-Schleife ist dabei wie folgt:

```
<?php
     for (Startwert; Bedingung; Schleifenschritt) {
          Anweisungen
     }
?>
```

Ein Beispiel wäre:

```
<?php
     for ($i=0; $i < 10; $i++) {
          echo "$i, ";
     }
?>
```

Ausführen Bei dem obigen Beispiel wird zuerst ein Startwert definiert, in diesem Fall wird die Variable $i auf den Wert 0 initialisiert. Die Bedingung der Schleife ist $i < 10 , d.h. die Anweisungen der Schleife werden solange durchlaufen wie diese Bedingung erfüllt ist. Der Schleifenschritt ist im obigen Fall $i++ . Diese Schritt wird nach jedem Schleifendurchlauf von PHP durchgeführt, d.h. im obigen Fall wird nach dem Schleifendurchlauf der Wert der Variable $i um 1 erhöht.

Aufgaben (den Rest machen wir am 7. Mai)

1. Wann sollte man POST und wann GET einsetzen?

2. Wann sollte man eine while-Schleife und wann eine for-Schleife einsetzen?

3. Lass den Nutzer eine Zahl und ein Wort eingeben. Als Ausgabe soll das Wort sooft ausgegeben werden, wie in der Zahl festgelegt. Ist die Zahl größer als 1000 soll stattdessen „Du spinnst wohl.“ ausgegeben werden.

4. Zahlenraten Teil 2:

Verwende 2 Seiten: raten.php und auswerten.php (bitte beide Dateien anlegen).

Verbessere das Spiel:

a) Verwende ein Formular mit POST zum Eingeben der Zahl → raten.php

b) Gib in auswerten.php aus, ob die Zahl richtig geraten wurde. Falls nicht, gibt einen Tipp (zu groß / zu klein)

c) Lass den Nutzer nochmal raten (Link auf die Datei raten.php) → das Spiel beginnt von vorn.

d) Zusatz: denk die eine Möglichkeit aus, die Anzahl der Versuche anzuzeigen
(Tipps gibt Herr Anske)