Introductory Programming With PHP

Agyapong William

CODE-S FOUNDATION

July 8, 2019

Outline I

Outline II

4. Reading Assignment: Operators
   - Basic Types of Operators

5. Reading Assignment: Decision Making
   - The IF Statements

Introduction

This course is meant to serve as a springboard needed to propel you into the very art of computer programming.
For specific emphasis, it will introduce you to PHP, a server side scripting language you can use to develop dynamic websites and web applications.

Installation & Configuration Process

- Before you can run PHP codes/programs on your local machine, you would need a server and the PHP language.
- Both can be installed at once using the xampp software. Simply download xampp and install it. During installation, accept all the default installation settings.
- After successful installation,open the xampp control panel and start the apache server by clicking the **start** button beside the apache module, then launch a browser and type "localhost" in the address bar and hit the enter key.
  This should take you to the xampp dashboard with a welcome message. Once you see this it means all is set for you to start writing your PHP programs.
- Next, you would need a text editor to write your PHP codes. In this course, we will be using **Visual Studio Code** which can be easily downloaded and installed.

Installation & configuration

- In your text editor, create a new file with the name 'welcome.php' and save it inside **htdocs** which is located here: Local Disk(C)-> xampp -> htdocs.

After saving the file, type the following:

```
<?php
        echo "<h2>Welcome to code-s</h2>";
?>
```

Now, with the apache running, launch a browser and type **localhost/welcome.php** and load the page. This should display the text "Welcome to code-s" in the browser window, indicating a successful PHP program execution.

Language Introduction

PHP is an amazing and popular language!
The acronym **PHP** is a recursive one, which stands for **Hypertext Preprocessor** - formerly **Personal Home Page**.

PHP is a widely-used, open source server-side scripting language best suited for use in web based applications. As it is a server side scripting language, its use requires a web server installed, configured and enabled with PHP.

To set the records straight, with PHP you're not limited to the development of websites. You can use PHP to develop a variety of tailor-made applications to comfortably run in a browser or have them deployed as traditional desktop applications.

SOME ATTRIBUTES OF PHP

- PHP is a scripting Language - Code written in PHP is not executed but interpreted by another program at runtime. No compilation hustles.
- PHP runs efficiently on the server side
- PHP is cross platform (i.e. it runs on various platforms such as Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP supports Object-Oriented Programming(OOP)- in PHP you can define your own reusable data structures called objects, making code/application development and maintainability a bliss
- PHP is open source which means it's free to download and use
- PHP has frameworks such as Laravel, CodeIgniter, etc. to support you on huge project development

SO WHAT CAN PHP DO?

- Generate dynamic page content
- Create, open, read, write, delete, and close files on the server
- Collect data from forms
- Read and write cookies (a small text file stored by a website on a user's hard drive)
- Communicate with databases
- Control user-access
- Encrypt data
- Detect the user's browser
- Generate XML documents, graphics, flash animations and pdf files.
- Run scripts from the command line

All these and more can be done with PHP!

Basic Language Concepts

- **PHP Tags**
  They provide the environment to write and run PHP codes. Any element found in between PHP tags are interpreted by the server. PHP offers different tags as alternatives. Below is the standard and recommended php tags
  - <?php *codes go here* ?> **NB:** This tag is usually aligned vertically as you shall see later on.
- **Outputting Results**
  The language constructs **echo()** and **print()** can be used to output PHP results. The shortcut syntax, <? = ...? >, provides an alternative structure.
- **Code Indentation**
  PHP does not implement any strict or fast rules on indentation. It's all up to you! But the rule of thumb here is to make your code as readable as possible.

- **Line Delimiter**
  The end of a PHP statement or line is indicated by a semicolon (**;**), which is known as the line delimiter. Failing to terminate multiple PHP statements will generate runtime errors. Remember to end every line with the semicolon.
- **Case Sensitivity**
  - User-defined function names and class names, built-in construct and keywords such as print, echo, while, if , class, etc, are case-insensitive. For instance, **if** and **IF** mean the same thing in PHP.
  - However, variables are case-sensitive. Meaning, $user and $USER refer to two different variables.

- **Whitespace Sensitivity**
  PHP is whitespace insensitive. It does not recognise multiple whitespace produced by the *SPACE BAR/TAB KEY* and the *RETURN KEY*. The reason being that PHP output is sent to the browser as ordinary HTML.
- **Comments**
  These are sections of codes that are ignored by PHP. We create a line comment by preceding the line with either '//' or '#' and a block/multilines comment with '/*comments */'

Data Types

PHP supports the following data types

Data Types

- **Scalar Types**
  - Numeric
    - Integer (Int)
    - Double / Float: Simply decimals
  - String: Series of characters like "hello world", "user1234", etc.
  - Boolean: True or false
- **Compound Types**
  - Array: Combination/list of items
  - Object: Instances of classes
- **Special Types**
  - Null: Represents nothing. The constant NULL is used.
  - Resource: References (e.g. a database connection)

The Notion of A Variable

A **variable** is a special space in memory defined to hold a value or data.
Variables in PHP are represented by a dollar sign followed by the name of the variable. The variable name is case-sensitive.
A valid variable name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.
The data type of a variable is not explicitly declared in PHP. Rather, it is decided at runtime by PHP depending on the context in which that variable is used. See an illustration below.

Illustration

```
<?php
        $score;
        $new_level;
        // you can assign values to variables like so:
        $name = 'Collins';
        $accountBalance = 100.00;
       // display output
        echo $name. ", your account balance is ". $accountBalance
?>
```

Output

Collins, your account balance is 100.00

Superglobals(predefined php array variables)

- $_SERVER
- $_SESSION
- $_COOKIE
- $_REQUEST
- $_GET
- $_POST
- $_FILES

We will discover more about these special built-in array variables later in the course

Changing Variable Data Type

```
<?php
    $token = "ID0010";
     echo gettype($token); //Original type is string

     // Using the settype() function
     settype($token, 'integer');
    echo gettype($token); //Now returns integer

    // Using casting
    $token = "ID0010";
     echo      gettype($token); //Original type is string

    $new_token = (integer)$token;
     echo gettype($new_token); //Now returns integer
?>
```

Constants

Constants are memory storage whose values cannot be changed or redefined.

They can only be defined by using **define('NAME', 'value')**. Use **get_defined_constants()** to access all defined constants including predefined ones.

NB: Only scalar data (boolean, integer, float and string) can be contained in constants.

Some Predefined Constants

- __LINE__ : Current line number of the file
- __FILE__ : Full path and filename of the file
- __CLASS__ : Class name
- __METHOD__ : Class method name
- __FUNCTION__ : Function name

Let's explore!

```
<?php
    echo 'File path: ' . __FILE__;
    echo 'Current line number: ' . __LINE__;

    $str = "some dummy text";
    echo gettype($str);   //what do you think gettype() does?
    var_dump($str);    //what do you think var_dump() does?

    $fruits = array('orange', 'banana', 'mango', 'berry');
    echo gettype($fruits);
    var_dump($fruits);
    print_r($fruits);
    //Can you tell what array() and print_r() are used for?

    echo ($_SERVER['PHP_SELF']);
    echo "<pre>", print_r($_SERVER) , "</pre>";
    echo "<pre>", print_r(get_defined_constants()) , "</pre>";
    // What's your guess?
?>
```

Reading Assignment: Operators

It is recommended that you read on these basic types of operators supported by php

Basic Types of Operators

- Arithmetic Operators
- Assignment Operators
- Logical Operators
- Comparison Operators
- Incrementing Operator
- Decrementing Operator
- Concatenation Operator

Reading Assignment: Decision Making

Read on the following php decision making constructs before our next meeting

- If Statements
- The Switch Statement
- The Ternary Statement