Contents

Basics

PHP is more of a template language compared to something like a scripting language. Although it is parsed like python, it does not "execute" all code it scans and will only execute specific PHP code.

A PHP file almost acts as a replacement for html files. Inside a php, writing html inside it like a normal html file will be ok. The only difference is this will have the extension *.php*.

A very unique thing about PHP is it *NEEDS* a server with PHP installed to run at all, unlike like something like JavaScript. If these requirements are not met then the server will not know how to parse the file and return the needed information.

When wanting to write the PHP code, it will require having inside. This is the only tag that is needed and all code will go in between them like:

```
<?php
    // php code here
    echo "Hello world";
    echo "<p>This was added through PHPs</p>";
?>
```

If the file contains ONLY PHP code, then only need to put is needed

The `echo` command will output some string(s) to the browser. This can be simple regular text strings or even things like html elements (this html elements will be added EXACTLY as put).

When it comes to where the PHP code is added, just like JavaScript, this fully depends where in the file this PHP code is added as that is where the stuff will be added.

To write a commit or multiline commit, it is the same as C/C++

Make sure to always put a semi-colon at the end of a line

All things listed before are case insensitive and anything not listed is case sensitive:

- Functions
- keywords *true* and *false*
- Class names
- logic statements
  - if
  - else-if
  - switch
  - while
  - do-while
  - for
  - foreach

White space does not have a special meaning in PHP, so doing the following are the same thing:

```
login(
    $username,
    $password, $age
);

login($username,$password, $age);
```

Builtin Functions

1. `define("const name", any value)` –> This will create a constant at runtime unlike using the normal *const* keyword which can only be done at compile time.
2. `var_dump(variable name)` –> This will return information about the variable passed in like value assigned to it and data type. This will not tell if it is a constant or not.
3. `die(optional single argument)` –> This is used to stop the execution of the program right there and then. This is an alias for another function called **`exit()`**. Calling either of these functions will do the same thing.
4. `array(any number of arguemnts)` –> This is used to make an array
5. `is_bool(boolean variable)` –> This is used to check if the variable is of a Boolean type. Will return 1 if it is and 0 otherwise.
6. `is_int(integer variable)` –> This is used to check of the variable is of an integer type. Will return 1 if it is and 0 otherwise.
7. `is_float(float variable)` –> This is used to check of the variable is of a float type. Will return 1 if it is and 0 otherwise.
8. `is_string(string variable)` –> This is used to check of the variable is of a string type. Will return 1 if it is and 0 otherwise.
9. `strlen(string variable)` –> This is used to get the length of a string
10. `unset(variable name)` –> This will set the variable to *null* regardless of its original value.
11. `is_null(variable name)` –> This is used to check of the variable is of a

null type. Will return true if it is and false otherwise.

12. “
13. “
14. “
15. “
16. “
17. “
18. “
19. “
20. “

Constants

Unlike normal languages that use a *const* keyword, PHP does need do that. Instead, it uses a function called **define()**. This function takes two arguments:

1. The name of the constant to have
2. The value the constant will be assigned. The value can be:

Can Use	Cannot Use
Number	Variables
Float	Objects
String	Function Calls
Array	
Other Constants	

Just like a variable, the naming sensitivity is the same like variables. However, the biggest difference is NOT using the dollar sign in front of the constant name. This can just be written out like normal. In fact even putting the $ in front of the constant variable will make it not work.

The naming convention for constants will have them be in all caps and if it is multiple words then seperate with _

There is a keyword called *const* that can be used, but this can only be assigned values that do not need to be calculated at runtime. If the value does need to be calculated at runtime then use the **define()** function. However, using *const* will make it defined at compile time.

```php
<?php
  define("Apple", 30); // Defines a constant called Apple

  echo $Apple; // this will not work and does not display the value 30
  echo Apple; // this will work and display the value 30
  define("APPLE_TOTAL", "Six");
echo $APPLE_TOTAL;
echo APPLE_TOTAL;

const NAME = "Gabriel Centeno";
 const NAME_FIRST = "Gabriel";
const NAME_LAST = "Centeno";
```

Control Flow

Just like other languages, PHP has the basic control flow like:

- if
- if else
- else
- switch

if statements

When it comes to this, it is basically the same as C.

```
<?php
  $x = 50;
  if ($x === 50){
    return "this is a good thing";
  }


  if ($x === 23){
    echo "yes";
  }
  else if ($x === 50){
      echo "no";
  }

  if ($x === 50){
    echo "yay";
  }
  else {
    echo "maybe";
  }

  if ($x === 30)
    echo "single thing";
```

> [!NOTE]

Just like in C, if there is only one statement will be executed there is no need to put the curly braces.