CS 328 PHP Coding Standards so far

- last modified: 2024-03-18
- For CS 328, you are expected to use **ONLY the following two types of tags** for your PHP embedded within a document:

  ```
  <?php
      ...
  ?>
  ```

  or

  ```
  <?= ... ?>
  ```

  - I'll call these **regular PHP tags** and **PHP expression tags**, respectively, below.
- For CS 328, we'll put the opening and closing parts of a **regular PHP tag** (`<?php` and `?>`) each on their own line, as shown above.
- Do the regular PHP tag's opening and closing parts need to line up? They can, as shown above, and when possible do so. However, see below for some possible exceptions.
  - Typically, indent the PHP statements within a regular PHP tag at least 3 spaces, and line then up.
  - BUT sometimes, for example when "jumping" in and out of static HTML, it is acceptable to line up its contents even with the regular PHP tag's parts.
  - AND I'll accept the opening and closing parts for regular PHP tags NOT lined-up with each other if they are instead lined up with the surrounding code, if the result maintains "overall" logic indentation in a pleasing way.
  - ...and `HERE` docs have their own required indentation idiosyncrasies!
  - (**the goal**: for your document including PHP to be **neat** and **readable**)
- It is **encouraged** to place **PHP expression tags** inline within HTML or document content. For example,

  ```
  <h1> Welcome to <?= $destination ?>! </h1>
  ```

- While the PHP Preprocessor may not enforce these, you are expected to:
  - **end** each statement within a **regular PHP tag** with a semicolon, but
  - **AVOID** putting a semicolon after the expression in a **PHP expression tag**.
- Unless you genuinely want the contents of a file to be able to be included more than once in a document (as for, perhaps, frequently-used HTML snippets), use `require_once` or `include_once` rather than `require` or `include`.
  - ...and choose between `require_once` or `include_once` based on whether the content being included SHOULD cause a fatal error if not available or not, respectively.
- PHP indentation guidelines: when you are using the style of control structures that include `{` and `}`:

  - the statement(s) within the body of the control structure should **not** be on the same line as the start of the control structure
  - the statement(s) within the body of the control structure should be indented by 3 or more spaces, and lined up
- You are encouraged to carefully "jump" in and out of PHP tags, avoiding or at least minimizing the number of `print` and `echo` statements.
  - (but if using `print` or `echo`, remember to include explicit newline characters so your resulting generated document does not have too-long lines or is otherwise "ugly".)
- You are expected to treat ALL user input as **UNTRUSTED** -- don't send it anywhere without trying to take steps to make sure that any attacks are detected and neutralized.
  - To guard against cross-site scripting, appropriately use PHP functions such as `htmlspecialchars`, `trim`, `strip_tags`, and `htmlentities`.
  - To guard against SQL injection, avoid dynamic SQL statements built using concatenation by, for example, use of bind variables, carefully-designed Oracle stored procedures, and carefully-designed Oracle stored functions.
  - (When you must use dynamic SQL statements built using concatenation, take special care to somehow check what is being concatenated.)