CS 328 PHP Coding Standards so far

- last modified: 2015-03-15
- 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 expression tags, respectively, below.
- For CS 328, we'll put the opening and closing tags of a regular php tag each on their own line, as shown above.
- Do the regular php tag's opening and closing tags 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 HTML5, it is acceptable to line up its contents even with the regular php tag,
  - AND I'll accept the opening and closing tags 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 expression tags inline within HTML5 or document content. For example,
  ```html
  <h1> Welcome to <?= $destination ?>! </h1>
  ```
- While the PHP Preprocessor may not enforce these, you are expected to:
  - end each statement in a regular php tag with a semicolon, but
  - AVOID putting a semicolon after the expression in an expression tag.
- Unless you genuinely want the contents of a file to be able to be included more than once in a document, 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: if you are using the style of control structures that include { and }:
  - the { and } are expected to each be on their own lines, lined up with the beginning of the control structure's first line
– the statement(s) within the { and } should be indented by 3 or more spaces within the { and }, and lined up
– (if you are using the alternative syntax for control structures,
  the labeled end should be lined up with the beginning of the control structure's first line, and
  the statements within the structure should be indented by at least three 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, try to appropriately use PHP functions such as strip_tags and htmlspecialchars.
  – To guard against SQL injection, try to avoid dynamic SQL statements built using concatenation by, for example, use of bind variables, carefully-designed Oracle stored procedure, 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.)