asgn02 - ch13: Functions

Objectives

- Write functions that return values.
- Develop program modularity
- Learn how to write functions that perform one task
- Write a function that calls another function
- Write fun

Setup

PHP Coding Assignment: Modular Functions and Single Responsibility Principle

- Create a folder named `asgn02-functions` in your `web182` folder.
- Copy the `chapter13` folder from your `web182-coursework-files` folder and paste it in your `web182/asgn02-functions` folder.
- Start up XAMPP or MAMP
- Start the Apache Webserver.

Assignment

Read chapter 13: Program Modularity – Working with Functions.

Functions are one of the first steps in creating modular programs. Modular code allows the programmer to 'separate their concerns'.

A function is a logical chunk of code that is separate from your main program.

Functions should only perform ***one*** task. Some of the advantages to writing modular programs are

- Makes programs easier to maintain.
- Keeps your main program from becoming too large and unwieldy.
- Makes writing programs easier by breaking down your program into separate modules.
- Makes your program easier to debug.

A main principle in programming is not to repeat yourself. If you see code that you keep writing repeatedly, it is usually a good candidate for a function.

Book exercises

Complete the following

- Review Questions
- All the fixits
- All the modifys
- Coding numbers 1, 3, 6 and 7. These are found at the end of the chapter.

Additional function exercises

These 2 exercises are **not** in your book.

Exercise: Shopping

Objective:

The goal of this assignment is to help you practice writing modular PHP code by creating functions that adhere to the Single Responsibility Principle. You will create a program that calculates the total price of items in a shopping cart, applies a discount if applicable, and outputs the final price.

Background:

A well-designed function should have a single purpose. This makes the code easier to understand, debug, and maintain. When each function is responsible for only one task, changes or updates to the code are less likely to introduce bugs in unrelated parts. By dividing your program into small, specific functions, you ensure modularity and reusability.

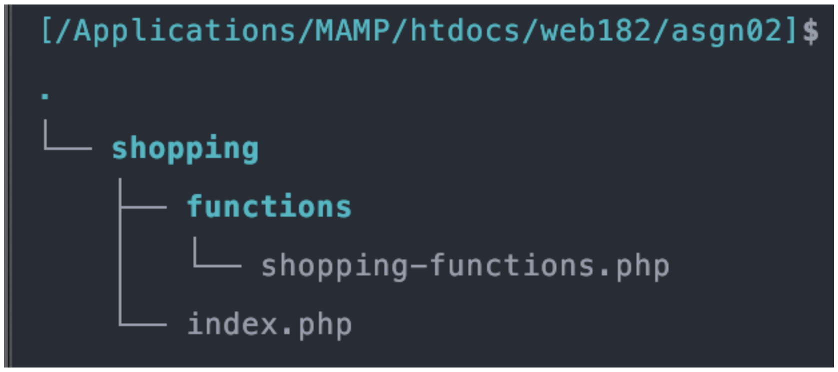
Instructions:

1. **Setup:**

   - Create a folder named `shopping` in your `asgn02` folder.
   - Create a file named `index.php` in your `shopping` folder
   - Create a folder named `functions` in your `shopping` folder.
   - Create a file named `shopping-functions.php` in your `functions` folder.

Here is what the file structure should look like.

Note: this image is from my Mac.



```
[/Applications/MAMP/htdocs/web182/asgn02]$
.
└── shopping
    ├── functions
    │   └── shopping-functions.php
    └── index.php
```

Store your functions in the `shopping-functions.php` file.

Use the `include()` function in your `index.php` file to reference the functions.

1. **Requirements:**

   - Write a function `calculateSubtotal()` that accepts an array of item prices and returns the sum of those prices.
   - Write a function `applyDiscount()` that accepts a subtotal and a discount percentage (e.g., 10 for 10%) and returns the discounted total.
   - Write a function `calculateTotal()` that calls both `calculateSubtotal()` and `applyDiscount()`. This function should:

     - Accept an array of item prices and a discount percentage as parameters.
     - Use `calculateSubtotal()` to get the subtotal.
     - Use `applyDiscount()` to calculate the total after applying the discount.
     - Return the final total.

   - Ensure the discount percentage is optional and defaults to `0` (no discount).

2. **Output:**

   - Your program should print the following to the console:
     - Subtotal before discount.
     - Discount applied (if any).
     - Final total price.

3. **Test Cases:**

   - Test your functions with the following inputs:
     - Item prices: `[10.00, 20.00, 30.00]`, Discount: `10%`
     - Item prices: `[50.00, 25.00]`, Discount: `0%`
     - Item prices: `[15.00, 5.00, 10.00]`, Discount: `20%`

Example Output:

```
Subtotal: $60.00
Discount Applied: 10%
Final Total: $54.00

Subtotal: $75.00
Discount Applied: 0%
Final Total: $75.00

Subtotal: $30.00
Discount Applied: 20%
Final Total: $24.00
```

Passing by Value and Reference in Functions**

Objective: So far we have passed arguments by value. A copy of the value is sent to the function). In this exercise you will learn how to pass an argument by the memory reference.

This tutorial demonstrates the concepts of passing arguments to a function by value and by reference.

Use an array of student scores and implement two functions: one that modifies a copy of the data (pass by value) and another that modifies the original data directly (pass by reference). This will help you understand how data is passed and modified in procedural PHP.

Tutorial: Passing by Value vs. Passing by Reference

In PHP, you can pass arguments to a function in two ways:

1. **Pass by Value:**

   - A copy of the variable is passed to the function.
   - Changes made inside the function do not affect the original variable.

   Example:

   ```
   function addOne($num) {
      $num++;
      return $num;
   }

   $original = 5;
   $result = addOne($original);

   echo $original; // Outputs: 5 (unchanged)
   echo $result;   // Outputs: 6
   ```

2. **Pass by Reference:**

   - The original variable is passed to the function.
   - Changes made inside the function directly affect the original variable.

   Example:

   ```
   function addOneByReference(&$num) {
      $num++ ;
   }

   $original = 5;
   addOneByReference($original);

   echo $original; // Outputs: 6 (changed)
   ```

1. Write a function named `addBonusPoints` that demonstrates **pass by value**:

   - Accepts an array of student scores as its argument.
   - Adds a fixed number of bonus points (e.g., 5 points) to each score.

- Returns the modified array without changing the original array.

2. Write another function named `addBonusPointsByReference` that demonstrates **pass by reference**:

   - Accepts an array of student scores by reference.
   - Adds the same fixed number of bonus points to each score.
   - Directly modifies the original array.

3. Create an array of student scores, where each score is an associative array with the following structure: `php [ 'name' => 'Student Name', 'score' => Numeric Score ]`

4. Call both functions separately and display the results to show the difference between pass by value and pass by reference.

Example

```php
<?php
// Define the function to add bonus points (pass by value)
function addBonusPoints($scores) {
    foreach ($scores as &$student) {
        $student['score'] += 5;
    }
    return $scores;
}

// Define the function to add bonus points (pass by reference)
function addBonusPointsByReference(&$scores) {
    foreach ($scores as &$student) {
        $student['score'] += 5;
    }
}

// Array of student scores
$students = [
    ['name' => 'Alice', 'score' => 85],
    ['name' => 'Bob', 'score' => 92],
    ['name' => 'Charlie', 'score' => 78],
    ['name' => 'Diana', 'score' => 88],
    ['name' => 'Ethan', 'score' => 95]
];

// Call the pass-by-value function
$updatedStudents = addBonusPoints($students);

// Call the pass-by-reference function
addBonusPointsByReference($students);

// Display the original and updated arrays
echo "Original Scores after Pass by Value:\n";
print_r($updatedStudents);

echo "\nOriginal Scores after Pass by Reference:\n";
print_r($students);
?>
```

Expected Output

After Pass by Value: ``` Array ([0] => Array ([name] => Alice [score] => 90)

```
[1] => Array
    (
        [name] => Bob
        [score] => 97
    )

[2] => Array
    (
        [name] => Charlie
        [score] => 83
    )

[3] => Array
    (
        [name] => Diana
        [score] => 93
    )

[4] => Array
    (
        [name] => Ethan
        [score] => 100
    )
```

) ```

After Pass by Reference: ``` Array ([0] => Array ([name] => Alice [score] => 95)

```
[1] => Array
    (
        [name] => Bob
        [score] => 102
    )

[2] => Array
    (
        [name] => Charlie
        [score] => 88
    )

[3] => Array
    (
        [name] => Diana
        [score] => 98
    )

[4] => Array
    (
        [name] => Ethan
        [score] => 105
    )
```

) ```

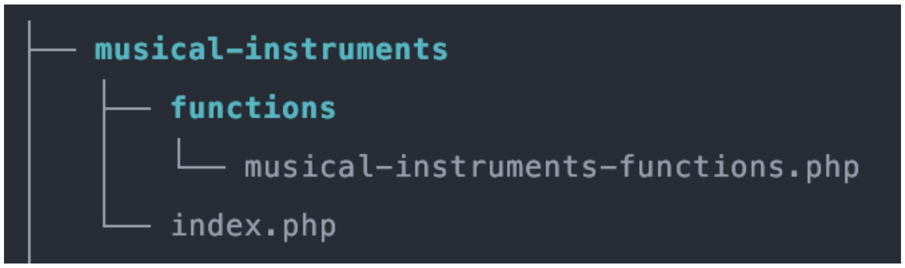
Exercise: Modify Instrument Prices

Objective: Practice working with passing by value and reference using an array of musical instruments and their prices.

Setup

- Create a folder named `musical-instruments` in your `asgn02` folder.
- Create a file named `index.php` in your `musical-instruments` folder.
- Create a folder named `functions` in your `musical-instruments` folder.
- Create a file named `musical-instruments-functions.php` in your `functions` folder.

Here is what the file structure should look like.



```
├── musical-instruments
│   ├── functions
│   │   └── musical-instruments-functions.php
│   └── index.php
```

Store your functions in the `value-reference-functions.php` file.

Use the `include()` function in your `index.php` file to reference the functions.

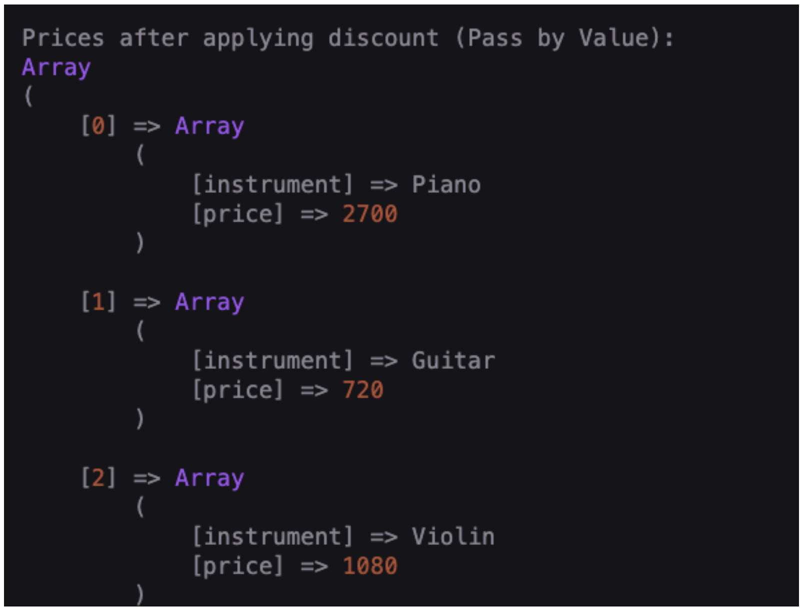
1. Use the `pass_by_value_and_reference.php` script as a guide.
2. Create an array of musical instruments, where each instrument is an associative array:
   `php [ ['instrument' => 'Piano', 'price' => 3000], ['instrument' => 'Guitar', 'price' => 800], ['instrument' => 'Violin', 'price' =>`
3. Write two functions:
   - One function (`applyDiscount`) that reduces the price of each instrument by a given percentage (pass by value).
   - Another function (`applyDiscountByReference`) that directly modifies the original array by applying the same discount (pass by reference).
4. Call both functions, passing the instrument array and a discount percentage (e.g., 10%).
5. Display the results to demonstrate the difference between passing by value and reference.

How to call the functions

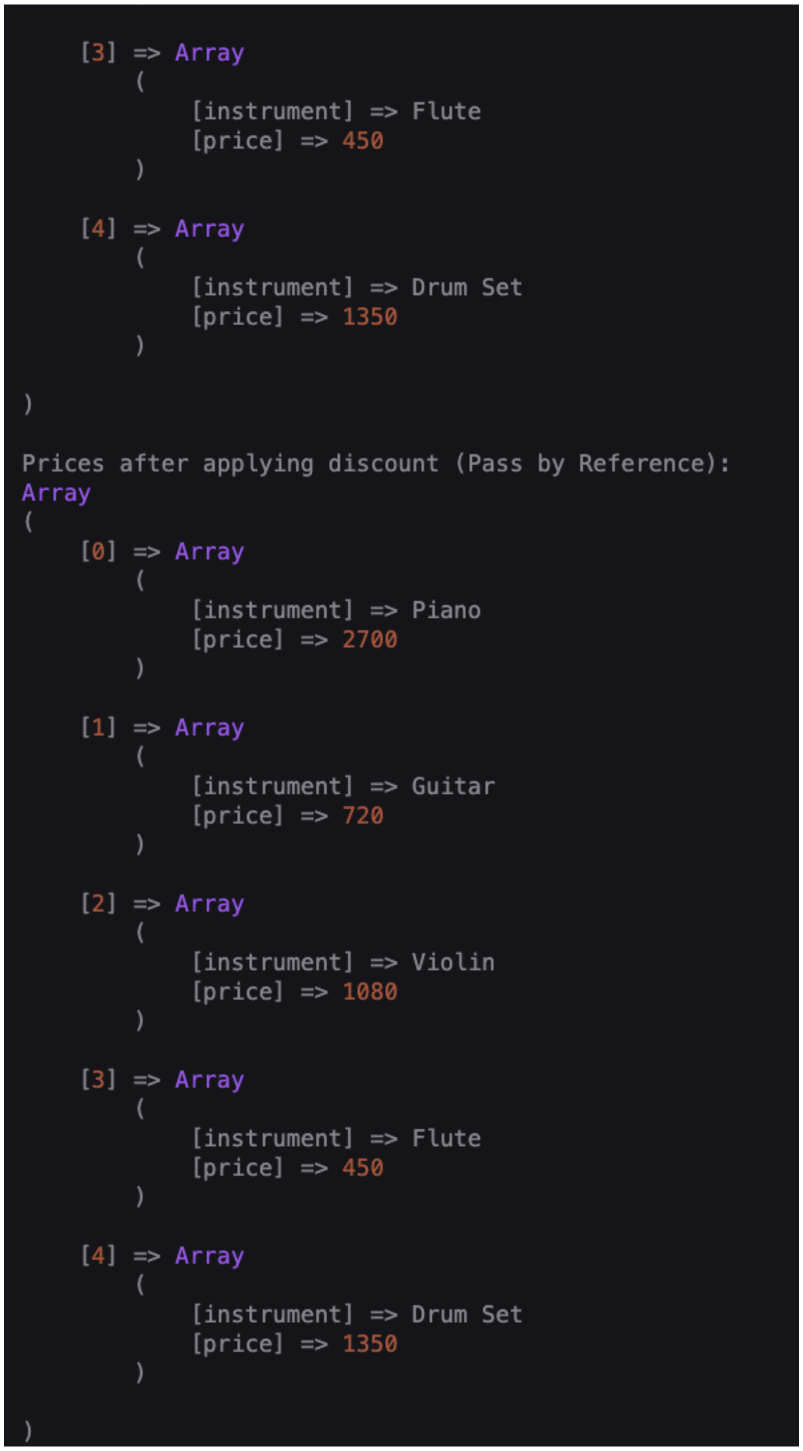
```php
// Apply discount (pass by value)
$discountedInstruments = applyDiscount($instruments, 10);

// Apply discount (pass by reference)
applyDiscountByReference($instruments, 10);

// Display the results
echo "Prices after applying discount (Pass by Value):\n";
print_r($discountedInstruments);

echo "\nPrices after applying discount (Pass by Reference):\n";
print_r($instruments);
```

What your output should look like

I am using the FireFox `var_dump` extenstion for this image. Chrome has the same extension but it is called `var_masterpiece`



```
Prices after applying discount (Pass by Value):
Array
(
    [0] => Array
        (
            [instrument] => Piano
            [price] => 2700
        )

    [1] => Array
        (
            [instrument] => Guitar
            [price] => 720
        )

    [2] => Array
        (
            [instrument] => Violin
            [price] => 1080
        )
```



```
    [3] => Array
        (
            [instrument] => Flute
            [price] => 450
        )

    [4] => Array
        (
            [instrument] => Drum Set
            [price] => 1350
        )

)

Prices after applying discount (Pass by Reference):
Array
(
    [0] => Array
        (
            [instrument] => Piano
            [price] => 2700
        )

    [1] => Array
        (
            [instrument] => Guitar
            [price] => 720
        )

    [2] => Array
        (
            [instrument] => Violin
            [price] => 1080
        )

    [3] => Array
        (
            [instrument] => Flute
            [price] => 450
        )

    [4] => Array
        (
            [instrument] => Drum Set
            [price] => 1350
        )

)
```

Submit

- Be sure you have completed the Review Questions.
- Zip your ch13 folder.
- Name the zipped file **yourLastName-yourFirstName- ch13.zip**