6DWEB Hands-On Test1

This 3-part activity marks as your hands-on test requirement for prelim. Sharing/Copying/GPT/AI codes will be marked zero.

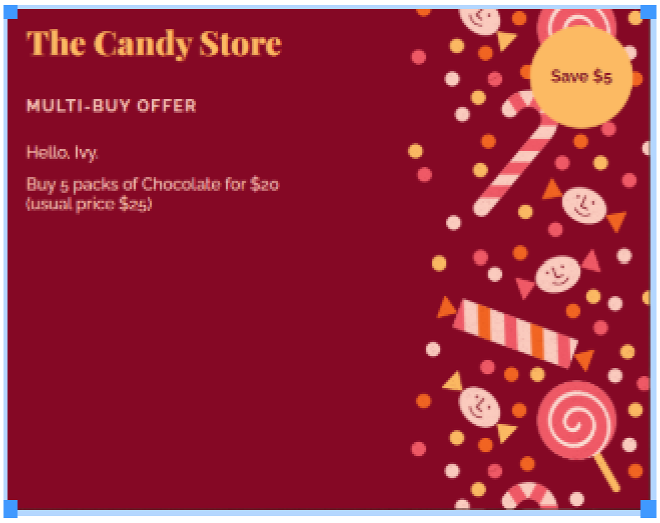
Once all files are downloaded, the instructor will disable the connection until you are ready to upload.

The source files provided are templates only. You will have to change elements (such as background and image [a must] to make it look uniquely created by you).

Note: *The term "product" mentioned throughout this activity means you decide your own product to showcase (except Candy). The files you downloaded will be modified as you go on.*

All steps provided below are must essential in your output/s.

Part 1:
A BASIC PHP PAGE: Processing and Displaying Data (Home)



The Candy Store

MULTI-BUY OFFER

Hello, Ivy.

Buy 5 packs of Chocolate for $20 (usual price $25)

Filename: index.php

The PHP file for this creates an HTML page that tells visitors about a discount that is available when they purchase multiple packs of your product.

You will have to:

- Store information in variables of arrays.
- Use the concatenation operator to join text in variables to create a personalized greeting for a visitor.
- Use arithmetic operators to perform calculations that determine the prices.
- Write new values that have been created by the PHP interpreter into the HTML contents of the page.

Additionally, if the values that are stored in the variables are updated then the page will automatically reflect the new products.

When you start writing PHP files, they are expected to contain a mixture of HTML and PHP code. It is a good practice to separate this code in the same file as much as possible:

- Start by using PHP to create the values that will be displayed in the HTML page and store those values in variables.
- Then, the lower part of the page will be focus on the HTML content. PHP code should only be used in this part of the page to display the values that have been stored in the variables.

To start the PHP part of the page:

1. The code starts by declaring a variable to hold the visitor's username. It is called **$user** because a variable name should always start with a dollar symbol, followed by the name that describes the type of data it holds.

2. A variable called **$pagbati** is declared to hold a greeting for the visitor. This issues the string operator to join the string Hello, and the name of the visitor.

3. A variable called **$offer** is created to hold the details of an item is on special offer. Its value is an array with four elements.

- The item of offer
- the quantity they must purchase
- The normal price per pack (without discount)
- The discounted price per pack

The first element, which describes the item on offer, uses a string data type. the other values are integers.

4. A variable called **$reg_price** is created its value is the cost of the items without discount. This is calculated by multiplying two of the values stored in the array: the quantity and the price.

5. A variable called $**offer_price** is created. It's value is the cost of the items with the discount applied. This is calculated by multiplying the quantity and the discounted price that were stored in the array.

6. A variable called **$saving** is created to hold the total saving for the customer. This is calculated by subtracting the value stored in the **$offer_price** variable (created in Step 5) from the value stored in **$usual_price** (created in Step 4).

The second half of the page will create the HTML that is sent back to the browser. It starts with the **HTML DOCTYPE** declaration (as usual). PHP is only used to write out values that were stored in variables in the previous steps:

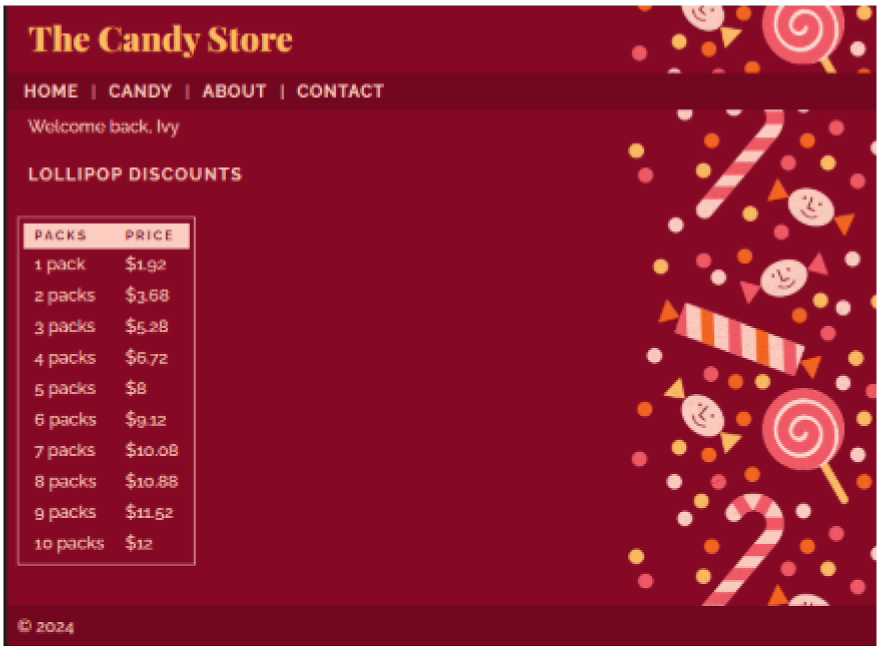
7. The greeting, which is the word **Hello** followed by the visitor's name, is written out to the page using the shorthand for the **echo** command.

8. The total saving, which is stored in the **$saving** variable (created in Step 6) is shown as a circle. CSS is used to place this circle in the top right corner of the browser window.

9. A paragraph explains the details of the offer. It shows the quantity of product the visitor has to buy, and the name of the product.

10. This is followed by the discounted price stored in **$offer_price** and the casual price in **$casual_price**.

Part 2: Showcasing Product's Prices and Discounts with Control Structures (Products)



The Candy Store

Welcome back, Ivy

LOLLIPOP DISCOUNTS

PACKS	PRICE
1 pack	$1.92
2 packs	$3.68
3 packs	$5.28
4 packs	$6.72
5 packs	$8
6 packs	$9.12
7 packs	$10.08
8 packs	$10.88
9 packs	$11.52
10 packs	$12

© 2024

Filename: product.php (name of your product as filename and will be the second page and in the navigation)

This next part of the activity shows a greeting, followed by any discounts that are applicable when a customer purchases multiple packs of your product. It will use a variety of the techniques that were introduced throughout Control Structures module.

- A variable stores the visitor's name.
- A conditional operator creates a greeting for the visitor.
- A for loop is used to create an indexed array that holds discounted prices that are on offer when customers buy multiple packs of your product.
- The header and footer of the page live in include files.
- A foreach loop is used to display to discounted prices from the array. For one pack there is a 4% discount, for two packs the discount is 8%, for three packs packs the discount is 12% and so on...
- A ternary operator ensures that the page displays the word pack when showing the price of a single pack of your product, or the word packs when showing the price of multiple packs.

The file starts by creating values that will be shown in the page and storing them in variables.

1. A variable called **$user** holds the user's name.

2. The **$greet** variable is initialized; it stores the message Hello which can be shown to anyone.

3. The condition of an if statement checks if the **$user** variable has a value.

4. If it does, the value in **$greet** is updated to hold a personal message using the visitor's name.

5. The name of the product is stored in **$product**.

6. The cost for one pack of it is stored in **$cost**.

7. A **for** loop creates an array to hold the prices of multiple packs of the product. The counter represents the number of packs of your product. In the parenthesis the:

- Counter is set to 1 (representing 1 pack of your product)
- Condition checks if the counter is less than or equal to 10 (representing 10 packs of product)
- The counter increments each time the loop runs.

Inside the loop:

8. The **$subtotal** variable holds the price of an individual pack of your product multiplied by the counter (which represents the current number of packs).

9. The **$discount** variable holds the discount when buying this number of packs of product. It is calculated by dividing the cost by 100, then multiplying that figure by the number if the counter multiplied by 4.

10. The **$totals** variable holds an array; the key is the current value in the counter (representing the number of packs) and the value is the price for this number of packs of your product minus the discount.

11. The for loop is closed.

In the part of the page that generates the HTML that is sent back to the browser:

12. The header for the page is included using the **require** keyword because it is needed in order for the rest on the page to display correctly. (You may **header.php** file from your practice activity previously but you will modify the contents to match to your ideal site created here).

13. The greeting is written into the page.

14. The name of the product is written into the page.

15. An HTML table is created, and column headings are added to the first row of the table.

16. A **foreach** loop is used to display the data that was stored in the array that was created in Steps 7-11. A new row of data is added to the table for each element in that array. In the parentheses, the:

- Array to use is stored in a variable called **$totals**
- Key is stored in a variable called **$quantity**
- Value is stored in a variable called **$price**

17. The key for this element of the array is written out (this is the number of packs of your product).

18. The text **pack** is written out. the condition of a ternary operator then checks if the value in **$quantity** is 1. If it is, nothing is added. If it is any value other than 1, an **s** is added (so it reads **packs** rather than the singular *pack*).

19. The price for this quantity of your product (with the discount) is handed out.

20. The **foreach** loop is closed.

21. The footer for the page is included using the include keyword (You may use your modified **footer.php** file from your practice activity previously adding Created by: YourFirstName Lastname).

Modify:

a. In Step 6, change the value in *$cost* to **10**.
b. In Step 7, update the loop to run **15** times, showing the prices for up to 15 packs of your product.

Part 3:
Putting it all together.

1. Part 2 has a navigation which you made using the include (header.php and footer.php enhanced). Apply it to Part 1 of this activity.

2. Combine the php pages created where:

- In your HOME link shows part 1.
- PRODUCT (previously CANDY) link shows part 2.
- Then navigate if all works. 2 links must change before doing submission/screenshot as proof.

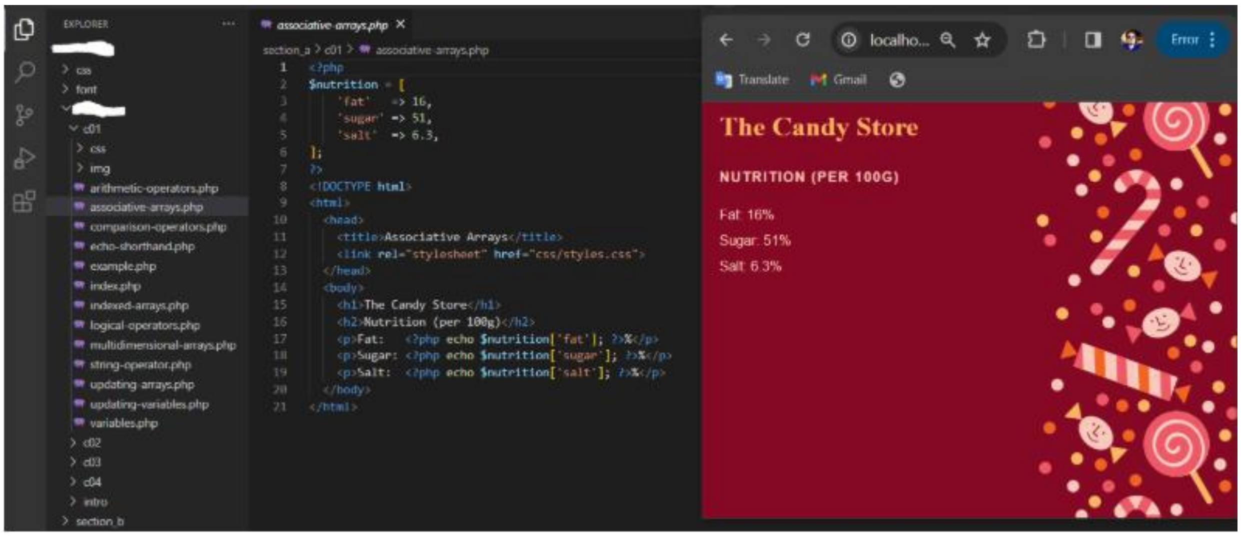
SUBMISSION:

Submit screenshots of your outputs where the readable codes and output + explorer on the left to check your file placement (Check the sample below). Follow filename format for Canvas upload:
a. Part 1: ***BasicPage.png*** *(or other img format)*
b. Part 2: ***Products.png*** *(or other img format)*

Canvas: Only the 2 files above are needed for canvas upload.

GitHub: Upload your output in a new GitHub repository. No upload before due time, no output.

Screenshot below is just a sample for your upload guidance showing explorer on the left, readable code and browser output.


```
EXPLORER
[redacted]
> css
> font
v [redacted]
  v c01
    > css
    > img
    arithmetic-operators.php
    associative-arrays.php
    comparison-operators.php
    echo-shorthand.php
    example.php
    index.php
    indexed-arrays.php
    logical-operators.php
    multidimensional-arrays.php
    string-operator.php
    updating-arrays.php
    updating-variables.php
    variables.php
  > c02
  > c03
  > c04
  > intro
> section_b
```

associative-arrays.php

section_a > c01 > associative-arrays.php

```php
<?php
$nutrition = [
    'fat'   => 16,
    'sugar' => 51,
    'salt'  => 6.3,
];
?>
<!DOCTYPE html>
<html>
  <head>
    <title>Associative Arrays</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Nutrition (per 100g)</h2>
    <p>Fat:   <?php echo $nutrition['fat']; ?>%</p>
    <p>Sugar: <?php echo $nutrition['sugar']; ?>%</p>
    <p>Salt:  <?php echo $nutrition['salt']; ?>%</p>
  </body>
</html>
```

```
The Candy Store

NUTRITION (PER 100G)

Fat: 16%
Sugar: 51%
Salt: 6.3%
```