1

VARIABILI, ESPRESSIONI E OPERATORI

Questo capitolo mostra come vengono memorizzati i dati nelle variabili, ovvero elementi che possono cambiare ogni volta che viene richiesta una pagina PHP, e come le espressioni e gli operatori possono operare sui valori delle variabili.

Le variabili impiegano un nome per rappresentare un valore, il quale può cambiare ogni volta che viene richiesta una pagina PHP.

- Il **nome** tende a descrivere il tipo di dati contenuto nella variabile.
- Il **valore** è il contenuto della variabile quando viene richiesta la pagina.

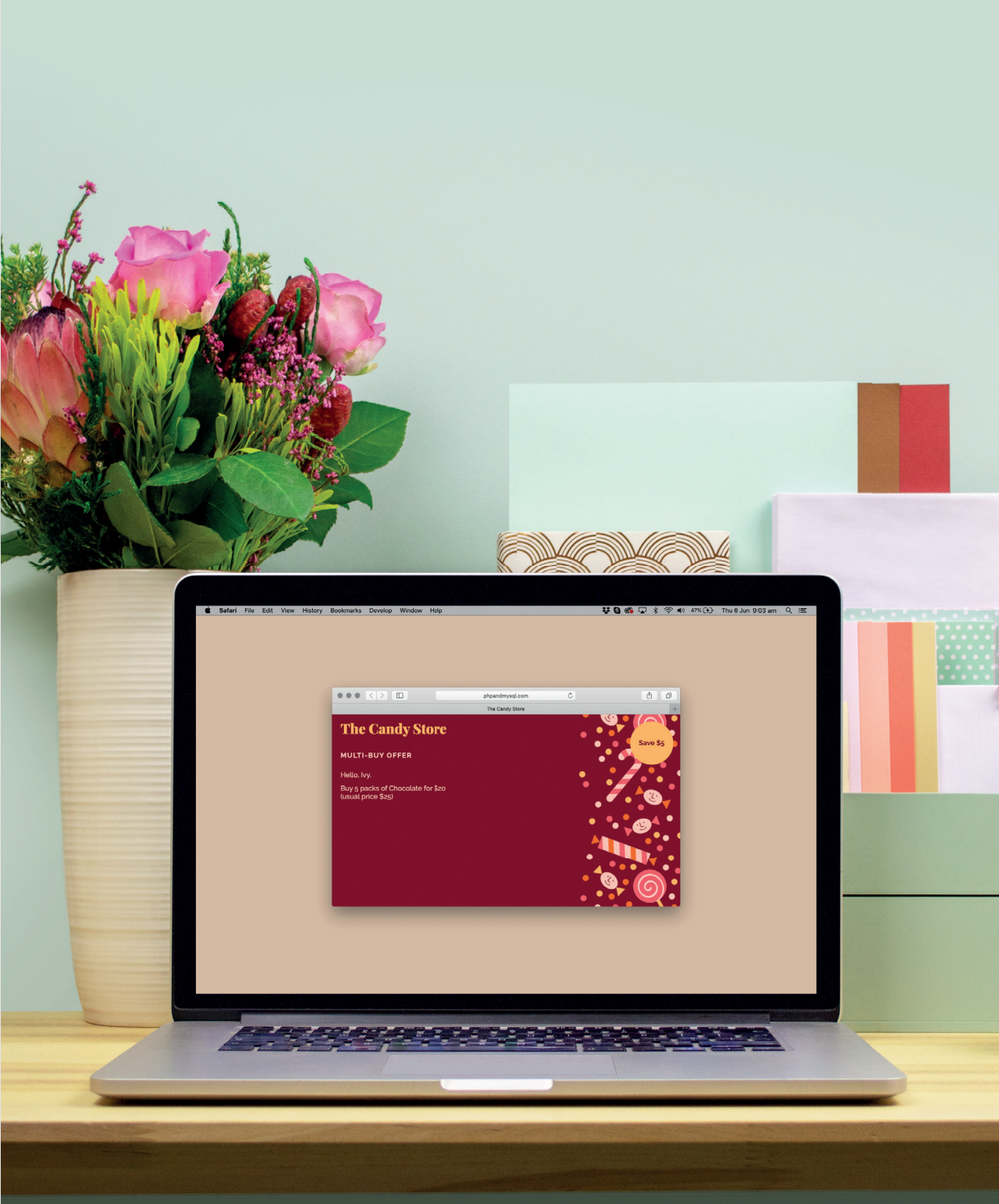
Al termine dell'esecuzione della pagina, dopo che l'HTML è stato inviato al browser, l'interprete PHP dimentica la variabile (quindi la volta successiva che la pagina viene eseguita, la variabile conterrà un valore differente).

PHP distingue i diversi tipi di valori che si possono memorizzare in una variabile (per esempio testo e numeri); essi sono noti come **tipi di dati**.

- Un frammento di testo è detto **stringa**.
- Un numero intero è detto **intero**.
- Un numero frazionario è rappresentato usando un numero in **virgola mobile**.
- Un valore che può essere solo vero (`true`) o falso (`false`) è detto **booleano**.
- Un elenco di nomi e valori correlati può essere memorizzato in un **array**.

Dopo aver introdotto le variabili, vedremo come le **espressioni** possono combinare più valori per creare un singolo valore. Per esempio, il testo contenuto in due variabili può essere concatenato per formare una frase, oppure un numero memorizzato in una variabile può essere moltiplicato con un numero contenuto in un'altra variabile.

Le espressioni impiegano **operatori** per creare un singolo valore. L'operatore +, per esempio, somma due valori, mentre l'operatore - sottrae un valore da un altro.


Safari File Edit View History Bookmarks Develop Window Help
Thu 6 Jun 9:03 am
phpandmysql.com
The Candy Store
The Candy Store
Save $5
MULTI-BUY OFFER
Hello, Ivy.
Buy 5 packs of Chocolate for $20
(usual price $25)

VARIABILI

Le variabili conservano dati che possono variare ogni volta che viene richiesta una pagina PHP. Si usa quindi un **nome** per rappresentare un **valore** che può cambiare.

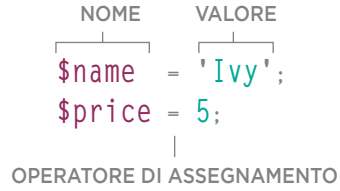
Per creare una variabile e memorizzarvi un valore, è necessario:

- un **nome** di variabile, che inizia con un segno di dollaro, seguito da una o più parole che descrivono il tipo di informazioni che la variabile può contenere;
- un **segno di uguale**, detto **operatore di assegnamento** in quanto assegna un valore al nome della variabile;
- il **valore** che volete inserire nella variabile.

Se la variabile contiene del testo, questo va posto tra virgolette. Si possono usare virgolette singole o doppie, ma senza scambiarle (non potete, per esempio, aprire una virgoletta singola e chiudere virgolette doppie).

Se la variabile conserva un numero o un valore booleano (`true` o `false`), non usano virgolette.

Quando si crea una variabile, le si dà un nome **dichiarandola**. Alla variabile viene poi **assegnato** un valore.



Dopo aver dichiarato una variabile e averle assegnato un valore, il nome della variabile può essere utilizzato nel codice PHP ovunque si desideri utilizzare il valore contenuto attualmente dalla variabile.

Quando l'interprete PHP incontra un nome di variabile, sostituisce al nome il valore in essa contenuto. Qui, il comando `echo` visualizza il valore contenuto nella variabile `$name`, dichiarata sopra.

COME CREARE E ACCEDERE ALLE VARIABILI

PHP section_a/c01/variables.php

```
<?php
① $name  = 'Ivy';
② $price = 5;
?>
<!DOCTYPE html>
<html>
  <head>
    <title>Variables</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
③   <h2>Welcome <?php echo $name; ?></h2>
    <p>The cost of your candy is
④     $<?php echo $price; ?> per pack.</p>
  </body>
</html>
```

OUTPUT

In questo capitolo, i valori vengono assegnati alle variabili nel codice PHP. Nei prossimi capitoli, i valori assegnati alle variabili verranno dai moduli HTML inviati dagli utenti, dai dati negli URL e dai database.

In questo esempio, all'inizio della pagina vengono create due variabili, cui viene assegnato un valore.

1. `$name` contiene il nome dell'attuale visitatore del sito. È un testo, quindi va scritto tra virgolette.

2. `$price` contiene il prezzo di un pacchetto di caramelle. È un numero, quindi il valore non va tra virgolette.

Di seguito, viene il codice HTML che viene inviato al browser dell'utente. Nell'HTML:

3. Il nome utente viene scritto nella pagina usando il comando `echo`.

4. Nella pagina viene scritto anche il costo delle caramelle.

PROVATE Nel Passo 1, modificate il valore della variabile `$name` con il vostro nome. Salvate il file e poi aggiornate la pagina nel browser. Il codice visualizzerà il vostro nome.

PROVATE Nel Passo 2, modificate il prezzo nel valore in 2. Salvate il file e poi aggiornate la pagina. Il codice visualizzerà il nuovo prezzo.

DARE UN NOME ALLE VARIABILI

Il nome di una variabile dovrebbe descrivere i dati che essa conserva. Utilizzate le seguenti regole per creare il nome di una variabile.

1

Iniziate con un segno di dollaro (`$`).

✓ `$greeting`
✗ `greeting`

2

Poi usate una lettera o un segno di sottolineatura (non un numero).

✓ `$greeting`
✗ `$2_greeting`

3

Quindi utilizzare qualsiasi combinazione di lettere a-z maiuscole e minuscole, numeri e trattini di sottilineatura (non sono consentiti trattini o punti).

✓ `$greeting_2`
✗ `$greeting-2`
✗ `$greeting.2`

NOTA `$this` ha un significato speciale. Da non usare come nome di variabile.

✗ `$this`

Se il nome delle variabili descrive i dati conservati nella variabile, il codice è più facile da leggere e comprendere.

Se servono più parole per descrivere i dati contenuti in una variabile, è prassi separare ogni parola con un segno di sottolineatura.

Nei nomi di variabile si distingue tra maiuscole e minuscole, quindi `$Score` e `$score` sono due variabili distinte. Tuttavia, evitate di creare due variabili che utilizzano la stessa parola e solo diverse combinazioni di lettere maiuscole/minuscole: è facile confonderle.

Tecnicamente, potete utilizzare caratteri di diversi set di caratteri (come quelli cinesi o cirillici), ma è considerata una buona pratica usare solo le lettere a-z, i numeri e i segni di sottolineatura, poiché sorgono alcuni problemi complicati nel supportare altri caratteri.

TIPI DI DATI SCALARI (DI BASE)

PHP distingue tre **tipi di dati scalari** che contengono testo, numeri e valori booleani.

TIPO DI DATI STRINGA

I programmatori chiamano un frammento di testo **stringa**. Il tipo di dati `string` può essere composto da lettere, numeri e altri caratteri, utilizzati per rappresentare testi.

```
$name = 'Ivy';
```

Le stringhe sono sempre racchiuse tra virgolette singole o doppie. La virgoletta di apertura deve corrispondere a quella di chiusura.

```
$name = 'Ivy';
$name = "Ivy";
$name = "Ivy';
$name = 'Ivy";
```

TIPI DI DATI NUMERICI

I tipi di dati numerici consentono di eseguire operazioni matematiche, come addizioni o moltiplicazioni, sui valori contenuti nelle variabili.

```
$price = 5;
```

I numeri non vanno scritti tra virgolette. Inserendo i numeri tra virgolette, verranno trattati come stringhe anziché come numeri.

PHP ha due tipi di dati numerici:

`int` rappresenta numeri interi, per esempio, 275.

`float` contiene numeri in virgola mobile, per esempio 2.75.

TIPO DI DATI BOOLEANO

Il tipo di dati `boolean` può assumere solo due valori: `true` o `false`. Questi valori sono comunemente usati nella maggior parte dei linguaggi di programmazione.

```
$logged_in = true;
```

`true` e `false` vanno scritti in minuscolo e non vanno racchiusi tra virgolette. Inizialmente, i valori booleani potrebbero sembrare un po' astratti, ma sono molte le cose che possono essere solo vere o false. Qualche esempio.

- L'utente ha fatto il login?
- Ha accettato i termini e le condizioni?
- Un prodotto è idoneo per la spedizione gratuita?

TIPO DI DATI NULL

PHP offre anche un tipo di dati `null`. Può assumere solo il valore `null`. Indica che per una variabile non è stato specificato alcun valore.

CAMBIO DI TIPO

Nelle pagine 60-61 vedremo come l'interprete PHP può convertire un valore da un tipo di dati a un altro, per esempio, una stringa in un numero.

AGGIORNARE IL VALORE DI UNA VARIABILE

Potete modificare o sovrascrivere il valore conservato in una variabile assegnandole un nuovo valore, un po' come avete fatto quando l'avete creata.

1. Qui viene **inizializzata** la variabile `$name`. Questo significa che viene dichiarata e le viene assegnato un valore iniziale, che verrà utilizzato se la variabile non viene poi modificata nella pagina.

Il valore iniziale è `Guest`, scritto tra virgolette in quanto è un testo.

2. Alla variabile `$name` viene poi assegnato il nuovo valore `Ivy`.

3. La variabile `$price` contiene il prezzo di un pacchetto di caramelle.

Poi viene il codice HTML che verrà inviato al browser dell'utente.

4. Il nome viene scritto nella pagina utilizzando il comando `echo`, che mostra il valore aggiornato che è stato assegnato alla variabile `$name` al Punto 2.

5. Scrive nella pagina il costo delle caramelle.

section_a/c01/updating-variables.php **PHP**

```
<?php
(1) $name  = 'Guest';
(2) $name  = 'Ivy';
(3) $price = 5;
?>
<!DOCTYPE html>
<html>
  <head>
    <title>Updating Variables</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
(4)  <h2>Welcome <?php echo $name; ?></h2>
    <p>The cost of your candy is
(5)     $<?php echo $price; ?> per pack.</p>
  </body>
</html>
```

OUTPUT

PROVATE Nel Passo 2, assegnate alla variabile `$name` il vostro nome. Salvate il file e aggiornate la pagina nel browser. Verrà visualizzato il vostro nome.

PROVATE Aggiungete una nuova riga dopo il Passo 2 e assegnate a `$name` un altro nome. Salvate il file e aggiornate la pagina. Verrà visualizzato questo nome.

ARRAY

Una variabile può anche contenere un **array**, ovvero una serie di valori correlati. Gli array sono **tipi di dati composti** perché possono conservare più valori.

Un array è come un contenitore di più variabili correlate. Un array è quindi costituito da più **elementi**. Così come una variabile usa un nome per rappresentare un valore, ogni elemento in un array ha:

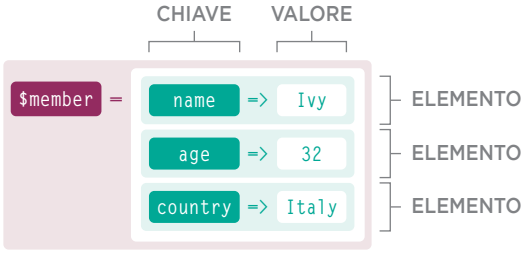
- una **chiave**, che agisce come nome della variabile;
- un **valore**, che rappresenta il contenuto dell'elemento.

PHP offre due tipi di array:

- negli **array associativi**, la chiave di ogni elemento è un nome che descrive i dati da esso conservati;
- negli **array indicizzati**, la chiave di ogni elemento è un numero, noto come **numero indice**.

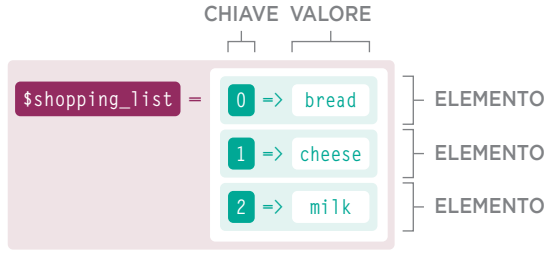
ARRAY ASSOCIATIVO

Questo array è progettato per contenere dati che rappresentano un utente iscritto al sito web. Ogni volta che viene utilizzato l'array, i nomi delle chiavi (che descrivono i dati conservati in ciascun elemento dell'array) restano gli stessi.



ARRAY INDICIZZATO

Questo array è progettato per contenere una lista della spesa. Elenchi come questo possono contenere un numero diverso di elementi a ogni utilizzo. La chiave non usa un nome ma un numero indice per descrivere ogni elemento: un intero che parte da 0.



In questi due esempi, ogni valore memorizzato nell'array è un tipo di dati scalare (un solo dato).

A pagina 44, potete vedere alcuni esempi di array in cui un elemento dell'array è a sua volta un array.

NOTA I numeri indice iniziano da 0, non da 1. Il primo elemento nell'elenco ha numero indice 0. Il secondo elemento è identificato dal numero indice 1 e così via. Il numero indice viene spesso utilizzato per descrivere l'ordine degli elementi nell'elenco.

ARRAY ASSOCIATIVI

Per creare un array associativo, assegnate a ciascun elemento (o oggetto) dell'array una **chiave** che ne descriva il contenuto.

Per memorizzare in una variabile un array associativo utilizzate:

- un nome di variabile che descrive l'insieme dei valori che saranno contenuti nell'array;
- l'operatore di assegnamento;
- le parentesi quadre per creare l'array.

All'interno delle parentesi quadre o tonde, utilizzate:

- il nome della chiave racchiusa tra virgolette;
- l'operatore doppia freccia =>;
- il valore dell'elemento (le stringhe vanno racchiuse tra virgolette; i numeri e i valori booleani no);
- una virgola dopo ogni elemento.

```
VARIABILE     CREA L'ARRAY
    |           |
$member = [
    'name'    => 'Ivy',
    'age'     => 32,
    'country' => 'Italy',
];      |       |       |
      CHIAVE OPERATORE  VALORE
```

È possibile creare un array associativo anche utilizzando la sintassi mostrata di seguito, con la parola `array` seguita da parentesi tonde (al posto di quelle quadre).

```
$member = array(
  'name'    => 'Ivy',
  'age'     => 32,
  'country' => 'Italy',
);
```

Per accedere a un elemento in un array associativo, utilizzate:

- il nome della variabile che contiene l'array;
- le parentesi quadre e le virgolette;
- la chiave per l'elemento che volete recuperare.

```
VARIABILE  CHIAVE
    |        |
 $member['name'];
```

CREARE E ACCEDERE AD ARRAY ASSOCIATIVI

PHP section_a/c01/associative-arrays.php

```php
<?php
$nutrition = [
    'fat'   => 16,
    'sugar' => 51,
    'salt'  => 6.3,
];
?>
<!DOCTYPE html>
<html>
  <head>…</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>
```

OUTPUT

1. In questo esempio, viene creato un array associativo, memorizzato nella variabile `$nutrition`.

L'array si trova tra parentesi quadre. Ha tre elementi (ogni elemento è una coppia chiave/valore). L'operatore `=>` assegna un valore a ogni chiave.

2. Per visualizzare i dati conservati nell'array usate:

- il comando `echo` per indicare che il valore che segue deve essere scritto nella pagina web;
- il nome della variabile array;
- le parentesi quadre e le virgolette che contengono il nome della chiave cui accedere.

Per esempio, per scrivere nella pagina il contenuto di `sugar` usate: `echo $nutrition['sugar'];`

PROVATE Passo 1: modificate i valori dell'array. Assegnate:
- a `fat` il valore `42`;
- a `sugar` il valore `60`;
- a `salt` il valore `3.5`.

Salvate e aggiornate la pagina per visualizzare i valori aggiornati.

PROVATE Nel Passo 1, aggiungete un elemento all'array. Usate la chiave `protein` e assegnatele il valore `2.6`. Poi, nel Passo 2, visualizzate il valore di `protein`.

ARRAY INDICIZZATI

Quando create un array, se non indicate una chiave per ogni elemento, l'interprete PHP gli assegnerà un numero, detto numero indice. I **numeri indice** iniziano da zero (0), non da uno (1).

Per memorizzare un array indicizzato in una variabile, usate:

- un nome di variabile che descrive l'insieme di valori che saranno contenuti nell'array;
- l'operatore di assegnamento;
- le parentesi quadre per creare l'array.

All'interno delle parentesi usate:

- l'elenco dei valori che conterrà l'array (le stringhe vanno racchiuse tra virgolette, i numeri e i booleani no);
- una virgola dopo ogni valore.

A ogni elemento verrà assegnato un numero indice.

Sopra, a `bread` verrebbe assegnato il numero indice `0`, a `cheese` `1` e a `milk` `2`. I numeri indice sono spesso usati per indicare l'ordine degli elementi dell'array.

È possibile creare un array indicizzato anche con la sintassi mostrata di seguito: con la parola `array` seguita da parentesi tonde (invece che quadre).

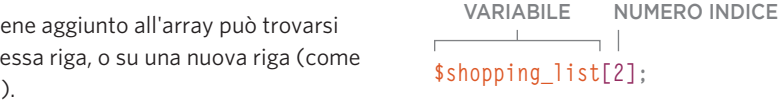
```
$shopping_list = array('bread',
                       'cheese',
                       'milk');
```

Ogni valore che viene aggiunto all'array può trovarsi di seguito, sulla stessa riga, o su una nuova riga (come nel caso mostrato).

Per accedere agli elementi di un array indicizzato, utilizzate:

- il nome della variabile che contiene l'array;
- seguito da parentesi quadre (senza virgolette);
- il numero indice dell'elemento cui volete accedere (tra parentesi quadre).

Il codice seguente chiede il terzo elemento dell'array, quindi in questo esempio il valore `milk`.



CREARE E ACCEDERE AD ARRAY INDICIZZATI

PHP section_a/c01/indexed-arrays.php

```php
<?php
$best_sellers = ['Chocolate', 'Mints', 'Fudge',
    'Bubble gum', 'Toffee', 'Jelly beans',];
?>
<!DOCTYPE html>
<html>
  <head>…</head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Best Sellers</h2>
    <ul>
      <li><?php echo $best_sellers[0]; ?></li>
      <li><?php echo $best_sellers[1]; ?></li>
      <li><?php echo $best_sellers[2]; ?></li>
    </ul>
  </body>
</html>
```

OUTPUT

1. Questo esempio inizia creando la variabile `$best_sellers`. Il suo valore è un array che contiene un elenco degli articoli più venduti sul sito web.

Questo array viene creato utilizzando parentesi quadre e gli elementi vengono aggiunti all'array all'interno di tali parentesi. Poiché gli elementi dell'array sono testi, vengono inseriti tra virgolette. I numeri e i valori booleani non vanno tra virgolette. Ogni elemento è seguito da una virgola.

2. I tre articoli più venduti vengono scritti sulla pagina:

- il comando `echo` indica il valore successivo da scrivere;
- seguito dal nome della variabile che contiene l'array;
- quindi dalle parentesi quadre che contengono il numero indice dell'elemento da recuperare. Ricordate che i numeri indice iniziano da 0 e non da 1.

PROVATE Nel Passo 1, aggiungete all'array l'elemento `Licorice` dopo `Fudge`. Nel Passo 2, aggiungete anche il quarto e il quinto elemento dall'array.

AGGIORNARE GLI ARRAY

Una volta creato l'array, potete aggiungervi nuovi elementi o aggiornare il valore di uno qualsiasi dei suoi elementi.

Per aggiornare un valore conservato in un array associativo, utilizzate:

- il nome della variabile che contiene l'array;
- seguito da parentesi quadre;
- quindi il nome della chiave racchiuso tra virgolette;
- l'operatore di assegnamento;
- il nuovo valore.

```
$member['name'] = 'Tom';
```

VARIABILE CHIAVE NUOVO VALORE

Per aggiornare un valore conservato in un array indicizzato, utilizzate:

- il nome della variabile che contiene l'array;
- seguito da parentesi quadre;
- il numero indice (non tra virgolette);
- l'operatore di assegnamento;
- il nuovo valore.

```
$shopping_list[2] = 'butter';
```

VARIABILE NUMERO INDICE NUOVO VALORE

Per aggiungere un nuovo elemento a un array associativo, fate come sopra, ma usate un nuovo nome di chiave (che non sia già stata utilizzata nell'array).

Le virgolette racchiudono il nome della chiave quando si tratta di una stringa, perché le virgolette indicano un tipo di dati stringa.

Vedremo come aggiungere elementi agli array indicizzati a p. 220. Il processo è diverso, perché potete specificare la posizione del nuovo elemento nell'array.

Le virgolette non vanno poste attorno ai numeri indice, perché i tipi di dati numerici non usano virgolette.

QUALE TIPO DI ARRAY UTILIZZARE

È meglio usare array associativi quando:

- sapete esattamente quali informazioni conterrà l'array; in tal modo potrete indicare un nome chiave per ciascuno degli elementi;
- avete bisogno che i singoli dati impieghino un nome ben preciso per la chiave.

È meglio usare array indicizzati quando:

- non conoscete il numero di dati che verranno memorizzati nell'array (i numeri indice crescono man mano che all'elenco vengono aggiunti elementi);
- volete memorizzare un elenco di valori in un ordine ben preciso.

MODIFICARE I VALORI CONSERVATI NEGLI ARRAY

PHP section_a/c01/updating-arrays.php

```php
<?php
$nutrition = [
    'fat'   => 38,
    'sugar' => 51,
    'salt'  => 0.25,
];
$nutrition['fat']   = 36;
$nutrition['fiber'] = 2.1;
?>
<!DOCTYPE html>
<html>
  <head>…</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>
    <p>Fiber: <?php echo $nutrition['fiber']; ?>%</p>
  </body>
</html>
```

OUTPUT

1. In questo esempio memorizziamo un array nella variabile `$nutrition`.

Le chiavi e i valori che compongono ogni elemento dell'array non devono necessariamente trovarsi su una nuova riga (come mostrato qui) ma questo ne rende più facili la lettura.

2. Il valore memorizzato per il contenuto di grassi (`fat`) viene aggiornato da 38 a 36.

3. All'array viene aggiunto un nuovo elemento. La chiave è `fiber` e il suo valore è `2.1`.

4. I valori nell'array vengono visualizzati sulla pagina.

PROVATE Dopo il Passo 3, aggiungete un'altra chiave per `protein` e assegnatele il valore `7.3`.

MEMORIZZARE UN ARRAY IN UN ALTRO ARRAY

Il valore di un elemento di un array può benissimo essere un altro array, per formare un **array multidimensionale**; una soluzione utile per rappresentare dati tabulari.

A volte è necessario memorizzare un insieme correlato di valori in un elemento di un array (per esempio, quando si rappresentano dati tradizionalmente tabulari). Considerate la tabella a destra: tre utenti iscritti, la loro età e il loro paese.

NAME	AGE	COUNTRY
Ivy	32	UK
Emi	24	Japan
Luke	47	USA

Ogni riga della tabella (ogni iscritto) può essere rappresentata con un elemento di un array indicizzato. Ogni elemento può quindi contenere un array associativo che memorizza il nome, l'età e il paese di ciascun iscritto.

I numeri indice per l'array indicizzato vengono assegnati automaticamente dall'interprete PHP. La virgola dopo ciascuno degli array associativi indica la fine del valore per quell'elemento.

```
$members = [
    ['name' => 'Ivy',  'age' => 32, 'country' => 'UK',],
    ['name' => 'Emi',  'age' => 24, 'country' => 'Japan',],
    ['name' => 'Luke', 'age' => 47, 'country' => 'USA',],
];
```

Per ottenere l'array che contiene i dati di Emi, usate:

- il nome della variabile che contiene l'array indicizzato;
- il numero indice dell'elemento cui desiderate accedere tra parentesi quadre (ricordando che gli array indicizzati iniziano da 0 e che i numeri non sono racchiusi tra virgolette).

```
$members[1];
```

Per ottenere l'età (`age`) di Luke, usate:

- il nome della variabile che contiene l'array indicizzato;
- il numero indice dell'elemento che contiene l'array di dati su Luke, tra parentesi quadre;
- la chiave dell'elemento cui desiderate accedere nell'array di Luke in una seconda serie di parentesi quadre (poiché la chiave è una stringa, servono le virgolette).

```
$members[2]['age'];
```

ARRAY MULTIDIMENSIONALI

PHP section_a/c01/multidimensional-arrays.php

```php
<?php
$offers = [
    ['name' => 'Toffee', 'price' => 5, 'stock' => 120,],
    ['name' => 'Mints',  'price' => 3, 'stock' => 66,],
    ['name' => 'Fudge',  'price' => 4, 'stock' => 97,],
];
?>
<!DOCTYPE html>
<html>
  <head>…</head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Offers</h2>
    <p><?php echo $offers[0]['name']; ?> -
      $<?php echo $offers[0]['price']; ?> </p>
    <p><?php echo $offers[1]['name']; ?> -
      $<?php echo $offers[1]['price']; ?> </p>
    <p><?php echo $offers[2]['name']; ?> -
      $<?php echo $offers[2]['price']; ?> </p>
  </body>
</html>
```

OUTPUT

1. Qui memorizziamo un array indicizzato nella variabile `$offers`.

Ogni elemento dell'array è un array associativo che contiene il nome, il prezzo e il livello di scorte di un articolo in offerta.

2. Visualizza il nome del primo prodotto (il numero indice del primo prodotto è 0).

3. Visualizza il prezzo del primo prodotto.

4. Visualizza il nome e il prezzo del secondo prodotto.

5. Visualizza il nome e il prezzo del terzo prodotto.

PROVATE Nel Passo 1, aggiungete all'array un altro prodotto di nome `Chocolate`. Impostate `price` a 2 e assegnategli un livello `stock` pari a 83. Quindi, dopo il Passo 5, visualizzate il nome e il prezzo del nuovo prodotto appena aggiunto.

Nel prossimo capitolo, vedremo come utilizzare un ciclo per scrivere il nome e il prezzo di ogni prodotto dell'array `$offers`, indipendentemente da quanti prodotti sono contenuti nell'array.

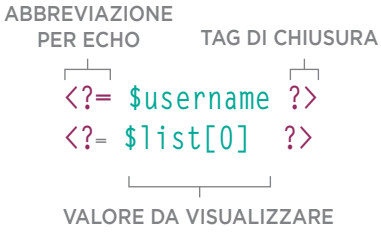
UN ECHO ABBREVIATO

Quando un blocco PHP viene utilizzato solo per scrivere un valore nel browser, potete utilizzare una forma abbreviata di `<?php echo ?>`.

Invece di scrivere `<?php echo $name; ?>` potete usare l'abbreviazione `<?= $name ?>` questa è l'unica volta che non è necessario usare il delimitatore di apertura `<?php`.

In tal modo *risparmiate*:

- le lettere `php` nel tag di apertura;
- il comando `echo`;
- il punto e virgola prima del tag di chiusura.



In molti degli esempi dei primi capitoli del libro, vedrete che ogni file PHP è composto da due parti:

- innanzitutto, il codice PHP che memorizza dei valori in variabili o array (potete anche svolgere delle attività con i dati in esse contenuti);
- poi, il codice HTML da inviare al browser; questa seconda parte della pagina mostra i valori che sono stati memorizzati nelle variabili utilizzando questa sintassi abbreviata.

Se iniziate ogni pagina creando i valori che la pagina dovrà visualizzare e memorizzandoli in variabili, otterrete una netta separazione tra il codice PHP da eseguire sul server e il codice HTML inviato al visitatore.

La seconda parte del file, in cui viene creata la pagina HTML, dovrebbe utilizzare la quantità minima possibile di codice PHP. Nei primi esempi, il codice PHP di questa parte della pagina visualizzerà solo i valori memorizzati nelle variabili.

USARE L'ABBREVIAZIONE DI ECHO

PHP section_a/c01/echo-shorthand.php

```php
<?php
① $name      = 'Ivy';
② $favorites = ['Chocolate', 'Toffee', 'Fudge',];
?>
<!DOCTYPE html>
<html>
  <head>
    <title>Echo Shorthand</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
③   <h2>Welcome <?= $name ?></h2>
    <p>Your favorite type of candy is:
④     <?= $favorites[0] ?>.</p>
  </body>
</html>
```

OUTPUT

Questo esempio crea due variabili e assegna loro dei valori nella parte superiore della pagina, prima dell'inizio del codice HTML.

1. `$name` contiene il nome di un utente iscritto al sito. Questo è testo, quindi va memorizzato tra virgolette.

2. `$favorites` contiene una serie di tipi di caramelle preferiti dagli utenti iscritti.

3. Il nome (`$name`) viene scritto nella pagina utilizzando l'abbreviazione del comando `echo`.

4. Il tipo di caramelle preferito dell'iscritto (`$favorites`) viene scritto sulla pagina utilizzando l'abbreviazione del comando `echo`.

PROVATE Nel Passo 1, modificate il valore contenuto nella variabile `$name` con il vostro nome. Nel Passo 2, aggiungete all'inizio dell'array il vostro tipo preferito di caramelle. Salvate il file e aggiornate la pagina del browser. Vedrete che il contenuto della pagina cambierà.

ESPRESSIONI E OPERATORI

Spesso vengono utilizzati due (o più) valori per crearne uno nuovo. Le **espressioni** sono costituite da uno o più costrutti che restituiscono un solo valore. Le espressioni usano **operatori** per creare un solo valore.

La matematica base (addizione, sottrazione, moltiplicazione e divisione) utilizza due valori per creare un nuovo valore. La seguente espressione moltiplica il numero 3 per 5 per creare il valore 15:

```
3 * 5
```

I programmatori dicono che le espressioni vengono **valutate** per produrre un singolo valore. Di seguito, il nuovo valore che viene creato viene poi memorizzato nella variabile `$total`:

```
$total = 3 * 5;
```

I simboli + - * / = sono detti **operatori**.

Potete unire due o più stringhe per creare un testo più lungo utilizzando un operatore per stringhe detto **operatore di concatenazione**. La seguente espressione unisce i valori `'Hi '` e `'Ivy'` per creare un'unica stringa.

```
$greeting = 'Hi ' . 'Ivy';
```

L'unione di queste due stringhe produce un unico valore, `Hi Ivy`, che viene memorizzato nella variabile `$greeting`.

Nel resto di questo capitolo tratteremo gli operatori introdotti nella pagina a destra.

OPERATORI ARITMETICI

pp. 50-51

Gli operatori aritmetici consentono di operare sui numeri, eseguendo attività come addizioni, sottrazioni, moltiplicazioni e divisioni.

Per esempio, se qualcuno sta acquistando 3 pacchetti di caramelle e ogni pacchetto costa $ 5, potreste utilizzare un operatore di moltiplicazione per calcolare il costo totale di quei tre pacchetti di caramelle.

OPERATORI PER STRINGHE

pp. 52-53

Gli operatori stringa consentono di operare sul testo. Esistono due operatori per stringhe che vengono utilizzati per unire diverse parti di testo in un'unica stringa.

Per esempio, se il nome di un utente iscritto è contenuto in una variabile e il cognome in una seconda variabile, potete unirle per creare il nome completo.

OPERATORI DI CONFRONTO

pp. 54-55 e p. 58

Come suggerisce il nome, gli operatori di confronto confrontano due valori e restituiscono un valore booleano `true` o `false`.

Per esempio, potreste confrontare i numeri 3 e 5 per vedere se:

- 3 è maggiore di 5 (`false`)
- 3 è uguale a 5 (`false`)
- 3 è minore di 5 (`true`)

Potete anche confrontare delle stringhe per vedere se un valore è maggiore o minore di un altro:

- `'Apple'` è maggiore di `'Banana'` (`false`)
- `'A'` è uguale a `'B'` (`false`)
- `'A'` è minore di `'B'` (`true`)

OPERATORI LOGICI

pp. 56-57 e p. 59

I tre operatori logici `and`, `or` e `not` operano sui due valori booleani `true` o `false`. Per comprenderne il funzionamento,considerate le seguenti due domande; a entrambe si può rispondere con `true` o `false`.

La temperatura è calda? È soleggiato?

- L'operatore `and` controlla se la temperatura è calda **e** (`and`) è soleggiato.
- L'operatore `or` controlla se la temperatura è calda **o** (`or`) è soleggiato.
- L'operatore `not` può verificare se la risposta a una sola di queste domande *non* è vera (true). Per esempio *non* è soleggiato?

Ciascuno di questi risultati produce un valore `true` o `false`.

OPERATORI ARITMETICI

PHP consente di utilizzare i seguenti operatori matematici.
Possono essere utilizzati con numeri o variabili che conservano numeri.

NOME	OPERATORE	USO	ESEMPIO	OUTPUT
Addizione	`+`	Somma due valori	`10 + 5`	`15`
Sottrazione	`-`	Sottrae un valore a un altro	`10 - 5`	`5`
Moltiplicazione	`*`	Moltiplica due valori (nota: si usa l'asterisco e non la lettera x)	`10 * 5`	`50`
Divisione	`/`	Divide due valori	`10 / 5`	`2`
Modulo	`%`	Divide due valori ma restituisce il resto	`10 % 3`	`1`
Esponente	`**`	Eleva un valore alla potenza dell'altro	`10 ** 5`	`100000`
Incremento	`++`	Incrementa un numero e restituisce il nuovo valore	`$i = 10; $i++;`	`11`
Decremento	`--`	Decrementa un numero e restituisce il nuovo valore	`$i = 10; $i--;`	`9`

ORDINE DI ESECUZIONE

Potete eseguire più operazioni aritmetiche in una singola espressione, ma è importante comprendere l'ordine con cui verrà calcolato il risultato:
la moltiplicazione e la divisione vengono eseguite *prima* dell'addizione e della sottrazione.

Questo può influenzare il risultato. In questo caso, per esempio, i numeri sono calcolati da sinistra a destra. Il risultato è 16:

```
$total = 2 + 4 + 10;
```

Nel seguente esempio, invece, il risultato è 42 (e non 60):

```
$total = 2 + 4 * 10;
```

Le parentesi consentono di indicare quale calcolo desiderate eseguire per primo, quindi quanto segue restituisce 60:

```
$total = (2 + 4) * 10;
```

Le parentesi indicano che il valore 2 deve essere sommato a 4 *prima* di essere moltiplicato per 10.

USARE GLI OPERATORI ARITMETICI

PHP section_a/c01/arithmetic-operators.php

```php
<?php
$items    = 3;
$cost     = 5;
$subtotal = $cost * $items;
$tax      = ($subtotal / 100) * 20;
$total    = $subtotal + $tax;
?>
<!DOCTYPE html>
  <html>
  <head>…</head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Shopping Cart</h2>
    <p>Items: <?= $items ?></p>
    <p>Cost per pack: $<?= $cost ?></p>
    <p>Subtotal: $<?= $subtotal ?></p>
    <p>Tax: $<?= $tax ?></p>
    <p>Total: $<?= $total ?></p>
  </body>
</html>
```

OUTPUT

Questo esempio mostra come utilizzare gli operatori matematici con i numeri per calcolare il costo di un ordine. Vengono, innanzitutto, create due variabili per memorizzare:

1. il numero totale di articoli ordinati (`$items`);
2. il costo di ogni pacchetto di caramelle (`$cost`).

Successivamente, vengono eseguiti i calcoli e, prima del codice HTML, i risultati vengono archiviati in variabili. Questo aiuta a separare il codice PHP dal contenuto HTML.

3. Per il costo dell'ordine si moltiplica il numero di articoli per il costo di un pacchetto di caramelle.

4. Vanno aggiunte le tasse, 20%. Per calcolarle, il subtotale viene diviso per 100 (le parentesi garantiscono che venga calcolato per primo) e il risultato viene poi moltiplicato per 20.

5. Infine, per trovare il costo totale, si somma l'imposta al subtotale.

6. I risultati memorizzati nelle variabili vengono quindi scritti nella pagina HTML.

PROVATE Modificate `$items` nel Passo 1 e `$cost` nel Passo 2.

OPERATORI PER STRINGHE

Per creare un singolo valore potrebbe essere necessario unire due o più stringhe. Il processo di unione di due o più stringhe è detto **concatenazione**.

OPERATORE DI CONCATENAZIONE

L'operatore di concatenazione è il simbolo di punto. Unisce il valore in una stringa al valore di un'altra. Nell'esempio seguente, la variabile `$name` conterrà la stringa `'Ivy Stone'`:

```
$forename = 'Ivy';
$surname  = 'Stone';
$name     = $forename . ' ' . $surname;
```

Notate che tra le variabili `$forename` e `$surname` viene aggiunto uno spazio; se non ci fosse, la variabile `$name` conterrebbe il valore `IvyStone`.

Potete concatenare tutte le stringhe che desiderate in un'unica istruzione, a condizione di porre l'operatore di concatenazione tra ogni stringa.

Potete unire stringhe contenute in variabili senza un operatore di concatenazione. Se un valore viene assegnato usando le doppie virgolette (al posto delle singole), l'interprete PHP sostituisce i nomi delle variabili tra virgolette con i valori che esse contengono. Qui `$name` conterrà il valore `Ivy Stone`.

```
$name = "$forename $surname";
```

OPERATORE DI CONCATENAZIONE E ASSEGNAMENTO

Se volete aggiungere del testo a una variabile esistente, potete usare l'operatore di concatenazione e assegnamento. È come una scorciatoia per aggiornare una stringa:

```
$greeting = 'Hello ';
$greeting .= 'Ivy';
```

Qui, la stringa `'Hello '` viene memorizzata nella variabile `$greeting`. Nella riga successiva, l'operatore di concatenazione e assegnamento aggiunge la stringa `'Ivy'` alla fine del valore contenuto nella variabile `$greeting`.

Ora, la variabile `$greeting` contiene il nuovo valore `'Hello Ivy'`. Come potete notare, utilizza una riga di codice in meno rispetto all'esempio a sinistra.

CONCATENARE LE STRINGHE

PHP section_a/c01/string-operator.php

```php
<?php
$prefix  = 'Thank you';
$name    = 'Ivy';
$message = $prefix . ', ' . $name;
?>
<!DOCTYPE html>
<html>
  <head>
    <title>String Operator</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2><?= $name ?>'s Order</h2>
    <p><?= $message ?></p>
  </body>
</html>
```

OUTPUT

Questo esempio mostra un messaggio personalizzato.

1. In primo luogo, creiamo una variabile `$prefix` per memorizzare il messaggio iniziale per l'utente, le parole 'Thank you'.

2. Crea una seconda variabile per memorizzare il nome dell'utente. Il nome della variabile è `$name` mentre quello dell'utente è `Ivy`.

3. Il messaggio personale viene creato concatenando tre valori insieme e memorizzando il nuovo valore della variabile `$message`:

- prima di tutto, viene aggiunto a `$message` il valore conservato in `$prefix`;
- poi, vengono aggiunti una virgola e uno spazio;
- infine, viene aggiunto il valore conservato in `$name`.

PROVATE Nel Passo 2, modificate il valore conservato in `$name` con il vostro nome.

PROVATE Nel Passo 3, assegnate il valore della variabile `$message` utilizzando le virgolette (e nessun operatore di concatenazione): `$message = "$prefix $name";`

OPERATORI DI CONFRONTO

Gli operatori di confronto consentono di confrontare due o più valori. Il risultato è un valore booleano `true` o `false`.

UGUALE A

Questo operatore confronta due valori per vedere se sono uguali.

`'Hello' == 'Hello'` restituisce `true`
perché *sono* la stessa stringa.
`'Hello' == 'Goodbye'` restituisce `false`
perché *non* sono la stessa stringa.

DIVERSO DA

Questi operatori confrontano due valori per vedere se *non* sono uguali.

`'Hello' != 'Hello'` restituisce `false`
perché *sono* la stessa stringa.
`'Hello' != 'Goodbye'` restituisce `true`
poiché *non* sono la stessa stringa.

Gli operatori sopra consentono all'interprete PHP di determinare se i due valori sono o meno equivalenti. Gli operatori sotto sono più rigorosi, perché controllano sia il valore *sia* il tipo dei dati.

Gli operatori sopra tratterebbero il numero 3 (un intero) come 3.0 (un float). Gli operatori sotto no. Nelle pagine 60-61 vedremo come 0 viene considerato come valore booleano `false` e 1 `true`.

IDENTICO A

Questo operatore confronta due valori per vedere se sono esattamente identici.

`'3' === 3` restituisce `false`
perché *non* sono lo stesso tipo di dati.
`'3' === '3'` restituisce `true`
perché *sono* lo stesso tipo di dati e lo stesso valore.

NON IDENTICO A

Questo operatore confronta due valori per vedere se non sono esattamente identici.

`3.0 !== 3` restituisce `true`
perché *non* sono lo stesso tipo di dati.
`3.0 !== 3.0` restituisce `false`
perché *sono* lo stesso tipo di dati e lo stesso valore.

Se usate `echo` per visualizzare il valore di un booleano sulla pagina, `true` visualizzerà `1` e `false` non visualizzerà nulla.

MINORE DI e MAGGIORE DI

`<` verifica che il valore a sinistra sia minore del valore a destra.

`4 < 3` restituisce `false` `3 < 4` restituisce `true`

`>` verifica che il valore a sinistra sia maggiore del valore a destra.

`z > a` restituisce `true` `a > z` restituisce `false`

MINORE O UGUALE A e MAGGIORE O UGUALE A

`<=` verifica che il valore a sinistra sia minore o uguale al valore a destra.

`4 <= 3` restituisce `false` `3 <= 4` restituisce `true`

`>=` verifica che il valore a sinistra sia maggiore o uguale al valore a destra.

`z >= a` restituisce `true` `z >= z` restituisce `true`

OPERATORE "SPACESHIP"

L'operatore "spaceship" confronta i valori a sinistra e a destra e restituisce:

`0` se i valori sono uguali
`1` se il valore a sinistra è maggiore
`-1` se il valore a destra è maggiore

Questo operatore è stato introdotto in PHP 7 e non funziona con le versioni precedenti di PHP.

`1 <=> 1` restituisce: `0`
`2 <=> 1` restituisce: `1`
`2 <=> 3` restituisce: `-1`

OPERATORI LOGICI

Gli operatori di confronto restituiscono un singolo valore, che può essere vero (`true`) o falso (`false`). Possono essere utilizzati con più operatori di confronto per confrontare, appunto, i risultati di più espressioni.

In questa riga di codice, sono presenti tre espressioni, ognuna delle quali restituirà un singolo valore `true` o `false`.

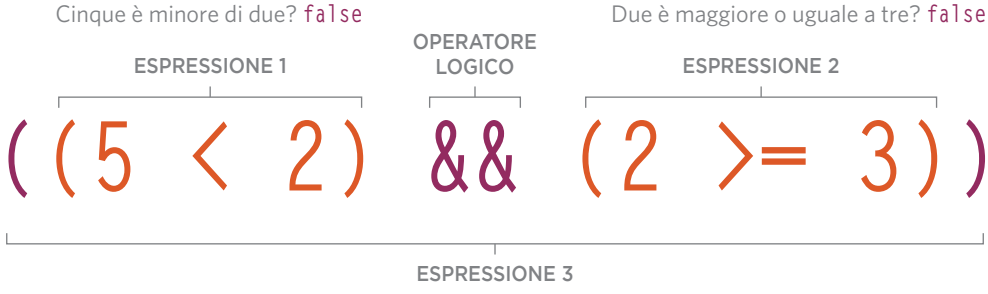
L'espressione 1 (a sinistra) e l'espressione 2 (a destra) utilizzano operatori di confronto ed entrambe restituiscono il valore `false`.

L'espressione 3 utilizza un operatore logico (non di confronto)

L'operatore logico `and` (`&&`) verifica se *entrambe* le espressioni (da entrambi i lati) restituiscono `true`. In questo caso, non lo fanno, quindi l'intera espressione viene valutata col valore `false`.

Le espressioni 1 e 2 vengono valutate prima della 3.

Ogni espressione è stata inserita nella propria coppia di parentesi. Questo aiuta a mostrare che il codice contenuto in ogni coppia di parentesi deve essere valutato per produrre un unico valore. Funziona anche senza parentesi, ma così è più comprensibile.



L'espressione 1 e l'espressione 2 restituiscono entrambe `true`? `false`

&&

AND LOGICO

Questo operatore verifica la verità di entrambe le condizioni.

```
((2 < 5) && (3 >= 2))
```
Restituisce il valore `true`

Solo se entrambe le espressioni restituiscono `true`, l'espressione restituisce `true`. Se una di queste è `false`, l'espressione restituisce `false`.

```
true  && true  restituisce true
true  && false restituisce false
false && true  restituisce false
false && false restituisce false
```

Potete utilizzare la parola riservata `and` invece di `&&`.

||

OR LOGICO

Questo operatore verifica la verità di almeno una condizione.

```
((2 < 5) || (2 < 1))
```
Restituisce il valore `true`

Se una delle due espressioni restituisce `true`, l'espressione restituisce `true`. Solo se entrambe le espressioni restituiscono `false`, l'espressione restituisce `false`.

```
true  || true  restituisce true
true  || false restituisce true
false || true  restituisce true
false || false restituisce false
```

Potete utilizzare la parola `or` invece di `||`.

!

NOT LOGICO

Questo operatore accetta un valore booleano e lo inverte.

```
!(2 < 1)
```
Restituisce il valore `true`

Nega un'espressione.
Se l'espressione (senza il simbolo `!`) è `false`, restituisce `true`.
Se l'affermazione è `true`, restituisce `false`.

```
!true  restituisce false
!false restituisce true
```

Non potete utilizzare la parola `not` al posto del punto esclamativo.

VALUTAZIONE A "CORTO CIRCUITO"

Le espressioni logiche vengono valutate da sinistra verso destra. Dopo che la prima espressione è stata valutata e l'interprete PHP riconosce l'operatore logico, potrebbe non essere necessario valutare la seconda condizione, come potete notare negli esempi a destra.

```
((5 < 2) && (2 >= 2))
```
Viene trovato un valore `false`.

Non ha senso controllare la seconda condizione, perché la prima è già `false`.

```
((2 < 5) || (2 >= 2))
```
Viene trovato un valore `true`.

Non ha senso controllare la seconda condizione perché sappiamo già che la prima è `true`.

USO DEGLI OPERATORI DI CONFRONTO

1. Crea tre variabili:

- la prima contiene il tipo di caramelle desiderate dal cliente;
- la seconda dice che ci sono 5 confezioni in magazzino;
- la terza mostra che il cliente vuole 8 confezioni.

2. Un operatore di confronto controlla se la quantità desiderata è minore o uguale alla quantità a magazzino. Il risultato è memorizzato nella variabile `$can_buy`.

3. Capita difficilmente di visualizzare un valore booleano (come in questo caso). È più probabile che il valore venga utilizzato in un'istruzione condizionale (vedi il prossimo capitolo). Ma è importante capire che cosa si ottiene provando a visualizzare un valore booleano come questo; se il valore è:

- `true`, verrà visualizzato 1;
- `false`, non verrà mostrato nulla.

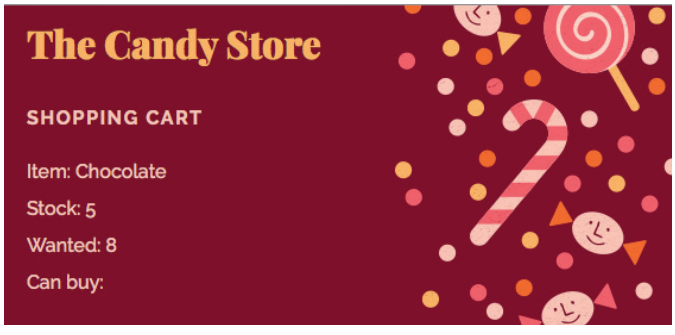
PROVATE Nel Passo 1, scambiate i valori in `$stock` e `$wanted`. Il valore di `$can_buy` cambierà.

A pagina 75, impareremo a mostrare messaggi diversi quando il risultato di un'espressione che utilizza un operatore di confronto è `true` o `false`.

section_a/c01/comparison-operators.php **PHP**

```
<?php
$item    = 'Chocolate';
$stock   = 5;                                    ①
$wanted  = 8;
$can_buy = ($wanted <= $stock);                  ②
?>
<!DOCTYPE html>
<html>
  <head>…</head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Shopping Cart</h2>
    <p>Item:    <?= $item ?></p>
    <p>Stock:   <?= $stock ?></p>
    <p>Wanted:  <?= $wanted ?></p>
    <p>Can buy: <?= $can_buy ?></p>               ③
  </body>
</html>
```

OUTPUT



USO DEGLI OPERATORI LOGICI

PHP section_a/c01/logical-operators.php

```php
<?php
$item    = 'Chocolate';
$stock   = 5;
(1) $wanted  = 3;
(2) $deliver = true;
(3) $can_buy = (($wanted <= $stock) && ($deliver == true));
?>
<!DOCTYPE html>
<html>
  <head>…</head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Shopping Cart</h2>
    <p>Item:    <?= $item ?></p>
    <p>Stock:   <?= $stock ?></p>
    <p>Ordered: <?= $wanted ?></p>
    <p>Can buy: <?= $can_buy ?></p>
  </body>
</html>
```

OUTPUT

Questo esempio si basa sull'esempio della pagina a sinistra.

1. Il cliente vuole solo 3 confezioni di caramelle.

2. Aggiunge la variabile `$deliver`; memorizza un valore booleano per indicare se è possibile effettuare o meno la consegna.

3. L'espressione utilizza due operatori di confronto:

- il primo controlla se ci sono sufficienti articoli in magazzino;
- il secondo controlla che l'articolo possa essere consegnato.

Un operatore logico `&&` verifica se entrambi gli operatori sono `true`. Se lo sono, il valore di `$can_buy` sarà `true` e la pagina visualizzerà il numero 1.

Se non risultano entrambi `true`, `$can_buy` conterrà un valore `false` e non verrà mostrato nulla.

PROVATE Nel Passo 1, scambiate i valori in `$stock` e `$wanted`. Il valore di `$can_buy` cambierà.

A pagina 75, imparerete a visualizzare messaggi diversi quando un'espressione restituisce `true` o `false`.

TYPE JUGGLING: CAMBIARE TIPO DI DATI

L'interprete PHP può convertire il tipo di dati di un valore.
Un comportamento noto come **type juggling**.

PHP è un linguaggio a **tipizzazione debole** perché, quando si crea una variabile, non è necessario specificare il tipo di dati del valore che conterrà. Sotto, la variabile `$title` contiene prima una stringa e poi un numero intero:

```
$title = 'Ten';  // Stringa
$title = 10;     // Intero
```

L'approccio di PHP può essere confrontato con linguaggi di programmazione a **tipizzazione forte** (come C++ o C#), che richiedono ai programmatori di specificare il tipo di dati di ciascuna variabile dichiarata.

Quando l'interprete PHP trova un valore che non utilizza il tipo di dati che si aspetta di ricevere, può provare a convertire il valore del tipo di dati previsto. Questo processo è detto **type juggling**.

Il cambio di tipo può creare confusione perché l'interprete PHP può generare risultati o errori sorprendenti. L'operatore di addizione seguente, per esempio, somma due valori. Il numero 1 è un intero, ma 2 è una stringa, perché è racchiusa tra virgolette.

```
$total =  1 + '2';
```

In questo caso, l'interprete PHP proverà automaticamente a convertire la stringa in un numero, in modo da poter eseguire l'addizione. Di conseguenza, la variabile `$total` conterrà effettivamente il numero 3.

Nella pagina a destra, potete vedere le regole che specificano come un valore viene convertito da un tipo di dati a un altro. Troverete alcuni esempi sul cambio di tipo all'indirizzo: `http://notes.re/php/type-juggling`.

Quando il tipo di dati di un valore viene cambiato, i programmatori dicono che il tipo di dati del valore viene **convertito** da un tipo a un altro.
Il type juggling è chiamato anche **casting implicito** perché è l'interprete PHP a eseguire automaticamente la conversione.

Quando un programmatore modifica esplicitamente il tipo di dati di un valore utilizzando il codice, si parla di **casting esplicito** perché all'interprete PHP è stato chiesto esplicitamente di modificare il tipo di dati.

NUMERI

Quando l'interprete PHP si aspetta due numeri, può eseguire un'operazione aritmetica su di essi.

Di seguito, potete vedere cosa succede quando:

- a un numero viene sommata una stringa;
- a un numero viene sommato un booleano.

NUMERO + STRINGA	OPERAZIONE	OUTPUT	DESCRIZIONE
`1 + '1'`	`1 + 1`	`2` (int)	La stringa contiene un intero e viene trattata come un intero.
`1 + '1.2'`	`1 + 1.2`	`2.2` (float)	La stringa contiene un float e viene trattata come un float.
`1 + '1.2e+3'`	`1 + 1200`	`1201` (float)	La stringa contiene un float con esponente e viene trattata come un float.
`1 + '5star'`	`1 + 5`	`6` (int)	La stringa contiene un intero seguito da caratteri. Il numero viene trattato come un intero e i caratteri vengono ignorati.
`1 + '3.5star'`	`1 + 3.5`	`4.5` (float)	La stringa contiene un float seguito da caratteri. Il numero viene trattato come un float e i caratteri vengono ignorati.
`1 + 'star9'`	`1 + 0`	`1` (int)	La stringa non inizia con un intero o un float. Viene trattata come il numero `0`.

NUMERO + BOOLEANO	OPERAZIONE	OUTPUT	DESCRIZIONE
`1 + true`	`1 + 1`	`2` (int)	Il valore booleano `true` viene trattato come l'intero `1`.
`1 + false`	`1 + 0`	`1` (int)	Il valore booleano `false` viene trattato come l'intero `0`.

STRINGHE

Quando l'interprete PHP tenta di concatenare due stringhe, seguirà queste regole.

Di seguito, potete vedere cosa succede quando PHP:

- concatena una stringa con un numero;
- concatena una stringa con un booleano.

STRINGA . NUMERO	TRATTATO COME	OUTPUT	DESCRIZIONE
`'Hi ' . 1`	`'Hi ' . '1'`	`Hi 1` (string)	L'intero viene trattato come una stringa.
`'Hi ' . 1.23`	`'Hi ' . '1.23'`	`Hi 1.23` (string)	Il float viene trattato come una stringa.

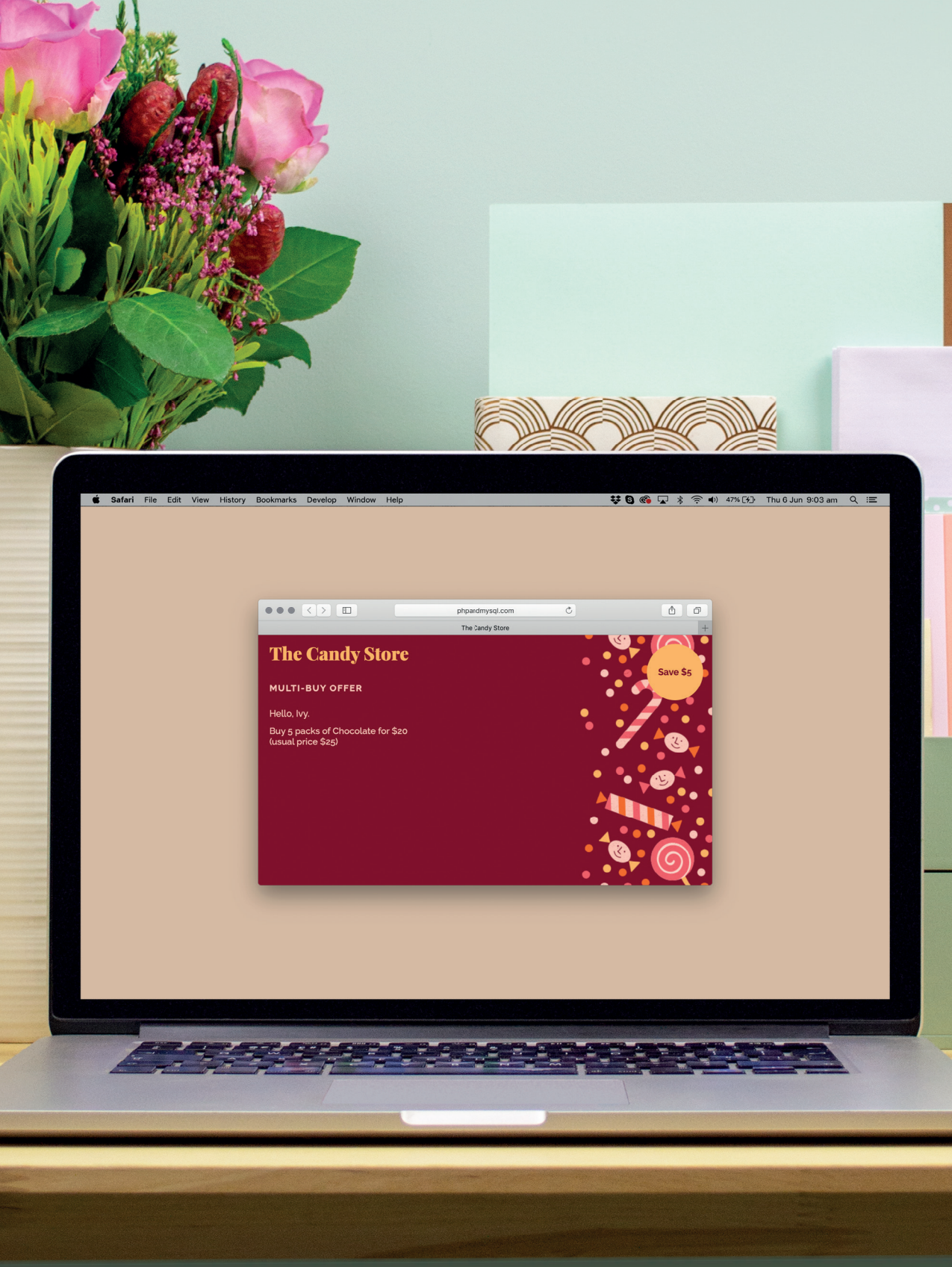
STRINGA . BOOLEANO	TRATTATO COME	OUTPUT	DESCRIZIONE
`'Hi ' . true`	`'Hi ' . '1'`	`Hi 1` (string)	Il booleano `true` viene considerato il numero intero `1`.
`'Hi ' . false`	`'Hi ' . ''`	`Hi` (string)	Il booleano `false` viene considerato una stringa vuota.

BOOLEANI

Quando l'interprete PHP si aspetta un valore booleano, tutti i valori mostrati nella tabella a destra verranno considerati `false`.

Qualsiasi altro valore viene considerato `true` (qualsiasi testo, un numero diverso da 0 o il valore booleano `true`).

VALORE	TIPO DI DATI	TRATTATO COME
`false`	Boolean	`false`
`0`	Integer	`false`
`0.0`	Float	`false`
`'0'`	stringa con valore 0	`false`
`''`	stringa vuota	`false`
`array[]`	array vuoto	`false`
`null`	Null	`false`


Safari File Edit View History Bookmarks Develop Window Help
47% Thu 6 Jun 9:03 am
phpandmysql.com
The Candy Store
The Candy Store
MULTI-BUY OFFER
Hello, Ivy.
Buy 5 packs of Chocolate for $20
(usual price $25)
Save $5

UNA SEMPLICE PAGINA PHP

Questo esempio riunisce molte delle tecniche trattate in questo capitolo.

Il file PHP crea una pagina HTML che informa gli utenti di uno sconto disponibile quando acquistano più confezioni di caramelle.

Vedrete come:

- memorizzare le informazioni in variabili e array;
- utilizzare l'operatore di concatenazione per unire il testo in variabili per creare un saluto personalizzato per l'utente;
- utilizzare gli operatori aritmetici per eseguire calcoli che determinano i prezzi visualizzati nella pagina;
- scrivere nuovi valori che sono stati creati dall'interprete PHP nel contenuto HTML della pagina.

Inoltre, se i valori conservati nelle variabili vengono aggiornati, la pagina rifletterà automaticamente i nuovi prodotti e prezzi.

ELABORARE E VISUALIZZARE I DATI

Quando si inizia a scrivere un file PHP, spesso si ha una combinazione di codice HTML e PHP. È buona norma separare il più possibile questo codice.

- Utilizzate codice PHP per creare i valori che verranno visualizzati nella pagina HTML e memorizzate i valori in variabili. Nell'esempio a destra, è la parte sopra la linea tratteggiata.

- La parte inferiore del listato può concentrarsi sui contenuti HTML. Il codice PHP deve essere utilizzato solo in questa parte della pagina per visualizzare i valori che sono stati memorizzati nelle variabili. Nell'esempio a destra, è la parte sotto la linea tratteggiata.

Esaminiamo il codice PHP all'inizio della pagina.

1. Questo esempio inizia dichiarando la variabile `$username` per il nome del visitatore. Tutti i nomi delle variabili iniziano con il segno di dollaro e dovrebbero descrivere il tipo di dati che contengono.

2. Dichiara la variabile `$greeting` che contiene la frase che accoglie il visitatore. Usa l'operatore per stringhe per unire la stringa `Hello` e il nome dell'utente.

3. Crea la variabile `$offer` per contenere i dettagli di un articolo in offerta speciale. Il suo valore è un array di quattro elementi:

- l'articolo in offerta;
- la quantità da acquistare;
- il prezzo per confezione (senza sconto);
- il prezzo scontato per confezione.

Il primo elemento, che descrive l'articolo in offerta, usa un tipo di dati stringa. Gli altri valori sono interi.

4. Crea la variabile `$usual_price`.
Il suo valore è il prezzo degli articoli senza sconto. Viene calcolato moltiplicando due dei valori memorizzati nell'array: la quantità e il prezzo.

5. Crea la variabile `$offer_price`.
Il suo valore è il prezzo degli articoli con lo sconto applicato. Viene calcolato moltiplicando la quantità e il prezzo scontato che sono stati memorizzati nell'array.

6. Crea la variabile `$saving` per contenere il risparmio totale per il cliente. Viene calcolata sottraendo il valore della variabile `$offer_price` (creata nel Passo 5) al valore di `$usual_price` (creata nel Passo 4).

La seconda metà della pagina (sotto la linea tratteggiata) crea l'HTML che verrà inviato al browser. Inizia dalla dichiarazione HTML `DOCTYPE`. Qui il codice PHP viene utilizzato solo per scrivere i valori che sono stati memorizzati nelle variabili nei passaggi precedenti.

7. Il saluto di benvenuto, ovvero la parola `Hello` seguita dal nome del visitatore, viene scritto sulla pagina utilizzando il comando `echo` abbreviato.

8. Il risparmio totale, conservato nella variabile `$saving` (creata nel Passo 6) è mostrato in un cerchio giallo. Viene impiegato del codice CSS per posizionare questo cerchio nell'angolo superiore destro della finestra del browser.

9. Un paragrafo spiega i dettagli dell'offerta. Mostra la quantità di dolci che il visitatore deve acquistare e il nome del dolce.

10. Segue il prezzo scontato in `$offer_price` e il prezzo normale in `$usual_price`.

PHP section_a/c01/example.php

```php
<?php
$username = 'Ivy';                                  // Variabile per il nome-utente

$greeting = 'Hello, ' . $username . '.';            // Saluto 'Hello, ' + nome-utente

$offer = [                                          // Crea l'array per l'offerta
    'item'     => 'Chocolate',                      // Articolo in offerta
    'qty'      => 5,                                // Quantità da acquistare
    'price'    => 5,                                // Prezzo di listino
    'discount' => 4,                                // Prezzo in offerta
];

$usual_price = $offer['qty'] * $offer['price'];     // Prezzo totale standard
$offer_price = $offer['qty'] * $offer['discount']; // Prezzo totale offerta
$saving      = $usual_price - $offer_price;         // Risparmio totale
?>
```

```html
<!DOCTYPE html>
<html>
  <head>
    <title>The Candy Store</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>

    <h2>Multi-buy Offer</h2>

    <p><?= $greeting ?></p>

    <p class="sticker">Save $<?= $saving ?></p>

    <p>Buy <?= $offer['qty'] ?> packs of <?= $offer['item'] ?>
       for $<?= $offer_price ?><br>(usual price $<?= $usual_price ?>)</p>
  </body>
</html>
```

PROVATE Nel Passo 1, sostituite il nome utente con il vostro nome.

Nel Passo 2, modificate il saluto di benvenuto con `Hi` (invece di `Hello`).

Nel Passo 3, aggiornate il numero di confezioni (`qty`) di caramelle nell'array `$offer` 3.

Nel Passo 3, aggiornate il prezzo delle caramelle a 6.

RIEPILOGO

VARIABILI, ESPRESSIONI E OPERATORI

- Le variabili conservano dati che possono variare ogni volta che viene eseguito uno script.
- I tipi di dati scalari contengono testo, numeri interi, numeri in virgola mobile e valori booleani `true` o `false`.
- Un array è un tipo di dati composto, utilizzato per memorizzare un insieme di valori correlati.
- Gli elementi in un array associativo hanno una chiave e un valore. Gli elementi in un array indicizzato hanno un numero indice e un valore.
- Gli operatori per stringhe uniscono (concatenano) il testo delle stringhe.
- Gli operatori matematici eseguono operazioni matematiche sui numeri.
- Gli operatori di confronto confrontano due valori per vedere se uno è uguale, maggiore o minore dell'altro.
- Gli operatori logici possono combinare i risultati di più espressioni utilizzando `and`, `or` e `not`.