Σύγκριση

- Με `==`, `!=`
- Ο τύπος των τιμών μετατρέπεται ώστε να γίνει η σύγκριση

- `1 == 1` → true
- `1 == 2` → false
- `0 == "hello"` → true
- `"1" == 1` → true

Σύγκριση

- Η σύγκριση αλφαριθμητικών γίνεται εύκολα και **σωστά**:

```
$a = “hello”;
if ( $a == “hello” ) {
    echo ‘$a is hello. ‘;
}
else {
    echo ‘$a is not hello. ‘;
}
```

if

```
if ( συνθήκη ) {
    σώμα 1
}
else if ( συνθήκη ) {
    σώμα 2
}
...
else {
    σώμα 3
}
```

if

- Εκτελεί κώδικα **υπό συνθήκη**
- Παρόμοιο με το if της C, C++, Java, …
- 1η συνθήκη **αληθής**;
  - Εκτέλεση κώδικα **σώματος** `if`
- Αλλιώς, 2η συνθήκη **αληθής**;
  - Εκτέλεση κώδικα **σώματος** `else if`
- …
- Όλες οι συνθήκες **ψευδείς**;
  - Εκτέλεση κώδικα **σώματος** `else`
- Τα `else if` και `else` είναι προαιρετικά

if

```
if ( $a == 5 ) {
    echo “Hello!”;
}
else {
    echo “Goodbye!”;
}
```

switch

```
switch ( παράσταση ) {
    case τιμή1:
        σώμα 1;
        break;
    case τιμή2:
        σώμα 1;
        break;
    ...
    default:
        εναλλακτικό σώμα
}
```

switch

- **Επιλέγει** ένα σώμα με βάση την τιμή μίας παράστασης
- Παρόμοιο με C, C++, Java, …
  - Τα cases **μπορούν** να είναι και **όχι σταθερές**
- Αν η παράσταση έχει τιμή1
  - Εκτέλεση σώματος 1
- Αν η παράσταση έχει τιμή2
  - Εκτέλεση σώματος2
- ...
- Αλλιώς
  - Εκτέλεση εναλλακτικού σώματος
- Το `default` είναι προαιρετικό

switch

```
switch ( $a ) {
    case 5:
        echo “a is 5”;
        break;
    case 10:
        echo “a is 10”;
        break;
    case 15:
        echo “a is 15”;
        break;
    default:
        echo “a is neither 5, nor 10, nor 15”;
}
```

switch

- Παράληψη του break οδηγεί σε **fall-through**
- Εκτελούνται τα σώματα που ακολουθούν μέχρι το επόμενο break

switch

```
$a = 10;
switch ( $a ) {
    case 5:
        echo “a is 5. ”;
    case 10:
        echo “a is 10. ”;
    case 15:
        echo “a is 15. ”;
    default:
        echo “a is neither 5, nor 10, nor 15”;
}
```

a is 10. a is 15. a is neither 5, nor 10, nor 15.

for

```
for ( αρχικοποίηση; συνθήκη; βήμα ) {
    σώμα
}
```

```
for ( αρχικοποίηση; συνθήκη; βήμα ) {
    σώμα
}
```

for

- Ίδιο σε C, C++, Java…
- Επαναλαμβάνει ένα σώμα σύμφωνα με κάποια συνθήκη
- Αρχικά τρέχει η **αρχικοποίηση**
- Αν **η συνθήκη** είναι **ψευδής**, τελειώσαμε
- Αν **η συνθήκη** είναι **αληθής**, τρέχει το **σώμα**
- Μετά το σώμα τρέχει το **βήμα**
- **Η συνθήκη ελέγχεται ξανά, κ.ό.κ.**

for

```
for ( $i = 0; $i < 5; ++$i ) {
    echo “Hello, world! “;
}
```

Hello, world! Hello, world! Hello, world!
Hello, world! Hello, world!

while

```
while ( συνθήκη ) {
    σώμα
}
```

```
while ( συνθήκη ) {
    σώμα
}
```

while

- Ίδιο σε C, C++, Java, …
- Επαναλαμβάνει ένα σώμα σύμφωνα με κάποια συνθήκη
- Αρχικά ελέγχεται η συνθήκη
- Αν **η συνθήκη** είναι **ψευδής**, τελειώσαμε
- Αν **η συνθήκη** είναι **αληθής**, τρέχει το **σώμα**
- **Η συνθήκη ελέγχεται ξανά, κ.ό.κ.**

while

```
$i = 0;
while ( $i < 5 ) {
    echo “Hello, world! “;
    ++$i;
}
```

Hello, world! Hello, world! Hello, world!
Hello, world! Hello, world!

do… while

```
do {
     σώμα
} while ( συνθήκη );
```

```
do {
    σώμα
} while ( συνθήκη );
```

do… while

- Ίδιο σε C, C++, Java, …
- Επαναλαμβάνει ένα σώμα σύμφωνα με κάποια συνθήκη
- Αρχικά τρέχει μία φορά το σώμα
- Στη συνέχεια ελέγχεται η συνθήκη
- Αν **η συνθήκη** είναι **ψευδής**, τελειώσαμε
- Αν **η συνθήκη** είναι **αληθής**, τρέχει το **σώμα**
- **Η συνθήκη ελέγχεται ξανά, κ.ό.κ.**

do… while

```
$i = 0;
do {
    echo “Hello, world! “;
    ++$i;
} while ( $i < 0 );
```

Hello, world!

break

- Ίδιο σε C, C++, Java, …
- Εμφανίζεται μέσα σε μία ροή ελέγχου
  - `for`, `foreach`, `while`, `do… while`, `switch`
- Διακόπτει την ροή και συνεχίζει αμέσως μετά
- Δεν γίνονται άλλες επαναλήψεις μετά το `break`

continue

- Ίδιο σε C, C++, Java, …
- Εμφανίζεται μέσα σε μία ροή επανάληψης
  - `for`, `foreach`, `while`, `do… while`
- Διακόπτει την ροή και συνεχίζει ελέγχοντας την συνθήκη
- Μπορεί να γίνουν και άλλες επαναλήψεις μετά το `continue`

Χειρισμός φορμών

- Για να πάρουμε δεδομένα από HTTP GET:
  - Μεταβλητή **$_GET**
  - **$_GET[“όνομα_παραμέτρου”]**
- Για να πάρουμε δεδομένα από HTTP POST:
  - Μεταβλητή **$_POST**
  - **$_POST[“όνομα_παραμέτρου”]**
- Ορίζονται αυτόματα από την PHP

Χειρισμός φορμών

input.html:

```
<form action="test.php" method="post">
     <input type="text" name="foo" />
     <input type="submit" value="Στείλε" />
</form>
```

Χειρισμός φορμών

test.php:

```
<p>
        Πληκτρολόγησες
        <?php
                echo $_POST[ 'foo' ];
        ?>!
</p>
```

Σχόλια

- `//` η υπόλοιπή γραμμή είναι σχόλιο
- Το πολύ 1 γραμμή

```
$a = 5; // assign $a to be 5
```

- `/*` τα περιεχόμενα είναι σχόλιο `*/`
- 1 ή περισσότερες γραμμές

Συναρτήσεις

```
function όνομα_συνάρτησης( ορίσματα ) {
     σώμα;
}
```

Συναρτήσεις

- Παρόμοιες με συναρτήσεις σε C, C++, Java, …
- Ορίζουν υπο-ρουτίνες που κάνουν συγκεκριμένη δουλειά
- Ορίζονται με την λέξη-κλειδί function
- Ακολουθεί το **όνομα** της συνάρτησης
- Ακολουθούν τα ονόματα των **ορισμάτων** σε ( ) χωρισμένα με κόμματα

Επιστροφή τιμής

- Οι συναρτήσεις **επιστρέφουν** τιμή με `return`
- Η τιμή επιστροφής χρησιμοποιείται όπου έγινε η κλήση
- Επιστροφή σημαίνει **τερματισμός** συνάρτησης
- Δεν ορίζουμε **τύπο** επιστροφής
- Δεν είναι υποχρεωτικό

Κλήση συναρτήσεων

- Καλούνται οπουδήποτε χρησιμοποιώντας το όνομα
- Ακολουθούν οι **τιμές** των ορισμάτων σε ( ) χωρισμένες με κόμματα
- Σειρά ορισμάτων έχει σημασία
  - Πρώτη τιμή → Πρώτο όρισμα
  - Δεύτερη τιμή → Δεύτερο όρισμα
  - κ.ό.κ.
- Κλήση χωρίς επιστροφή:

```
όνομα_συνάρτησης( τιμές_ορισμάτων );
```

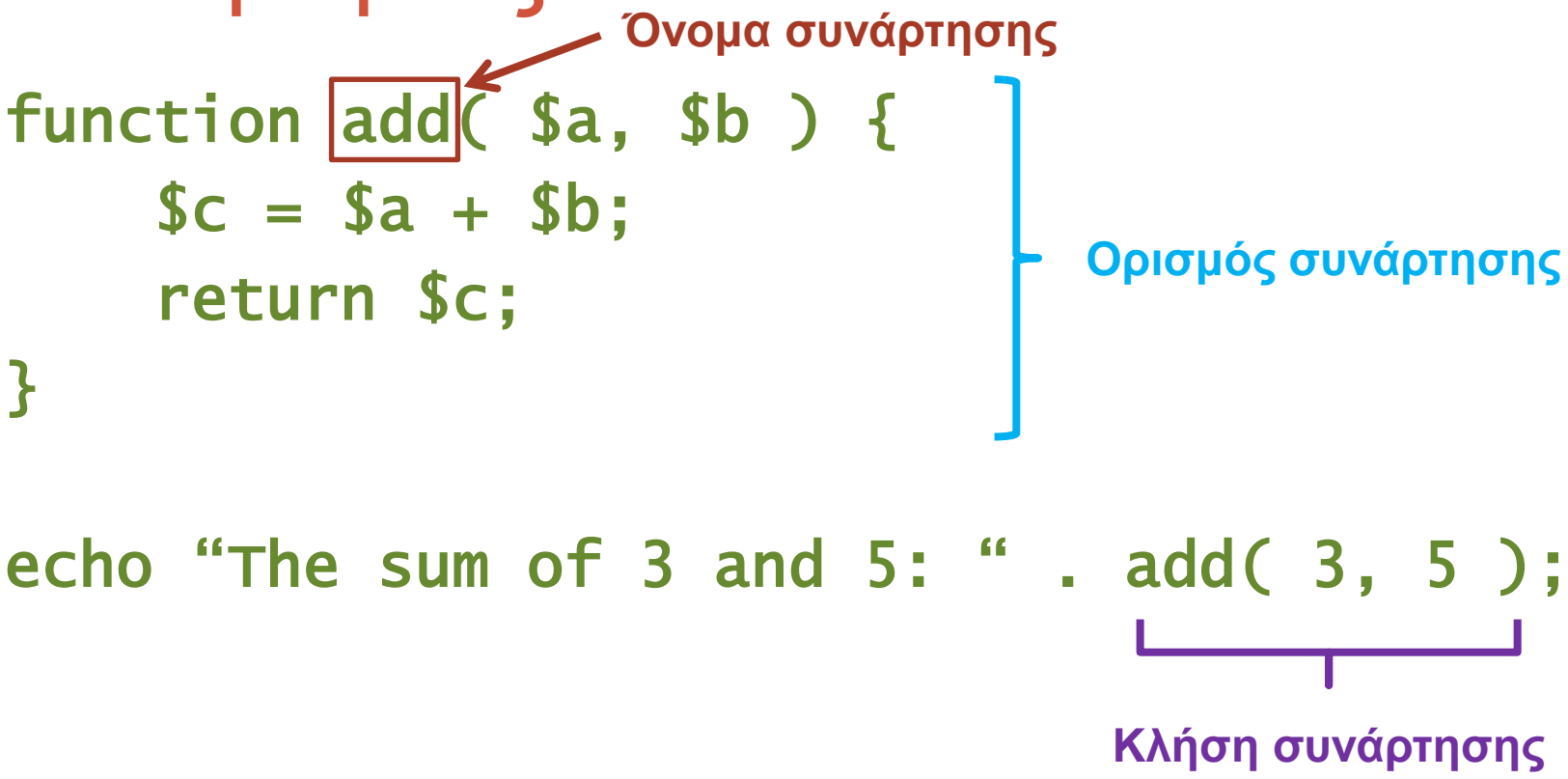
- Κλήση με επιστροφή:

```
$a = όνομα_συνάρτησης( τιμές_ορισμάτων );
```

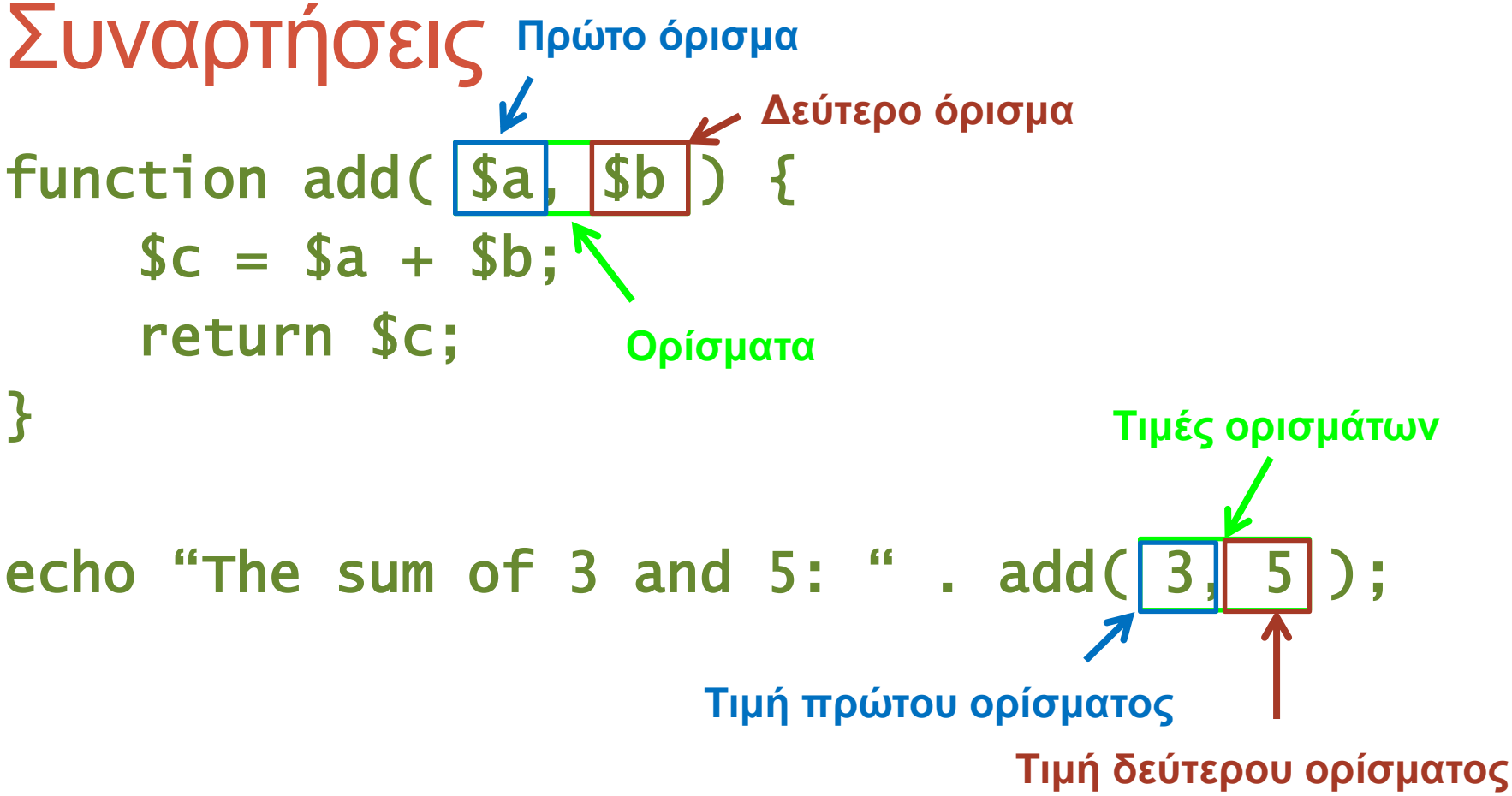
Ορίσματα

- **Δίνουν** πληροφορίες σε μία συνάρτηση
- Ακολουθούν ίδια ονοματολογία με μεταβλητές
- Αρχίζουν με $, ακολουθεί το **όνομα**
- Το όνομα...
  - Αρχίζει με γράμμα ή _
  - Περιέχει γράμματα, αριθμούς, _
  - Έχει ευαισθησία σε πεζά-κεφαλαία

Συναρτήσεις



Συναρτήσεις



Συναρτήσεις

```
function add( $a, $b ) {
    $c = $a + $b;
    return $c;
}

echo “The sum of 3 and 5: “ . add( 3, 5 );
```

Τιμή επιστροφής

Παίρνει αυτή τη θέση

Συναρτήσεις

```
function avg( $a, $b ) {
    $c = $a + $b;
    return $c / 2;
}

echo ‘The average of 3, 5: ‘ . avg( 3, 5 );
echo “\n”;
echo ‘The average of 1, 9: ‘ . avg( 1, 9 );
```

Συναρτήσεις

```
function is_prime( $a ) {
    for ( $i = 2; $i < $a; ++$i ) {
        if ( $a % $i == 0 ) {
            return false;
        }
    }
    return true;
}

if ( is_prime( 5 ) ) {
    echo “5 is a prime number.”;
}
```

Προαιρετικά ορίσματα

- Μπορούν να είναι:
- Τα **τελευταία** μίας συνάρτησης
- Όσα θέλουμε
- Ορίζουμε μία **προεπιλεγμένη** τιμή με το = μετά το όνομα του ορίσματος

Προαιρετικά ορίσματα

Όλα τα ορίσματα προαιρετικά

```
function makeCoffee
      ( $type = “frappe”, $milk = true ) {

      $str = “Making a cup of $type”;
      if ( $milk ) {
            $str .= ‘ with milk’;
      }
      $str .= “.\n”;
      return $str;
}

echo makeCoffee( “espresso” );
```

Προαιρετικά ορίσματα

2ο όρισμα προαιρετικό

```
function makeCoffee
      ( $type, $milk = true ) {

      $str = “Making a cup of $type”;
      if ( $milk ) {
            $str .= ‘ with milk’;
      }
      $str .= “.\n”;
      return $str;
}

echo makeCoffee( “espresso” );
```

Προαιρετικά ορίσματα

```
function makeCoffee
      ( $type = ‘frappe’, $milk ) {

      $str = “Making a cup of $type”;
      if ( $milk ) {
            $str .= ‘ with milk’;
      }
      $str .= “.\n”;
      return $str;
}

echo makeCoffee( “espresso” );
```