Funzioni

Una funzione è una porzione di codice che può essere eseguita più volte durante un programma.

Ogni funzione deve sempre ritornare un risultato o comunque mostrare qualcosa all'utente;

- Estrazione;
- Incapsularlo;
- Invocarlo.

Un esempio di una funzione che mostra qualcosa è il

```
$x = 6;
var_dump($x);
```

Si dividono in

- Built-in functions
- User Functions

1 - Built-in Functions

Le Built-in functions sono le funzioni predefinite del linguaggio che stiamo usando.

2 - User Functions

Sono le funzioni create dall'utente

2.1 - Dichiarazione ed Esecuzione

Cosa significa dichiarare una funzione?

1. Darle un nome
2. Esplicitare le istruzioni che compongono il corpo della funzione
3. Esplicitare i parametri che deve manipolare

Cosa significa eseguire una funzione?

1. Chiamarla utilizzando il nome scelto.

2.2 - Come si fa in pratica

```
//Dichiarazione
function nome_funzione()
{
  //istruzioni
}
//Chiamata
nome_funzione();

//Esempio

function saluta()
{
  echo "Ciao";
}
saluta();

//Oppure

saluta();

function saluta()
{
  echo "Ciao";
}
```

2.3 - Scope delle variabili

Scope inteso come **campo di applicazione, settore di competenza.**

Le variabili esterne alla funzione non sono visibili all'interno della funzione.

In PHP una **variabile normale** è visibile solo nella porzione di codice in cui è dichiarata.

```
$name = "Francesco";
function saluta()
{
  echo "Ciao $name";
  echo "Ciao Antonio";
}
saluta();
```

Discorso diverso invece sono le costanti:

```
const NAME = "Francesco";
function saluta()
{
  echo "Ciao Antonio";
  echo "Ciao" . NAME;
}
saluta();
```

2.4 - Parametri

Con parametri si intende l'insieme dei dati che la funzione deve manipolare, sono posizionali.

2.4.1 - Passaggio dei Parametri per Valore

```
$name = "Francesco";
function saluta($nome)
{
  echo "Ciao $nome";
}
saluta($name);
// Oppure
```

```
$num1 = 1;
$num2 = 2;

function sum($a, $b) //$a e $b Sono parametri Formali
{
  echo $a + $b;
}

sum($num1 , $num2);//Sono dett parametri Attuali
```

Altro Esempio:

```
$num1 = 1;
$num2 = 3;

function sum($a, $b) //$a e $b Sono parametri Formali
{
  echo $a + $b;
}

//sum($num1 , $num2);//Sono dett parametri Attuali
sum(1,3);
```

2.4.2 - Passaggio di valori per riferimento

Il passaggio per riferimento avviene quando passiamo non una copia del valore ma la locazione di memoria in cui è conservata la variabile.

Per passare una variabile per riferimento, utilizziamo il simbolo **&** prima del nome del parametro formale.

```
$x = 5;
```

```
function sum(&$a, &$b)
{
  echo $a + $b;
}

sum($x, $y);
```

2.5 - Differenze tra Valore e Riferimento

Per valore non si modifica il valore attuale:

```
$x = 11;
function increment($elemento)
{
  $elemento++;
}
echo "Prima della funzione x: $x\n";
increment($x);
echo "Dopo la funzione x: $x\n";
```

Per riferimento invece avremo:

```
$x = 11;
function increment(&$elemento)
{
  $elemento++;
}
echo "Prima della funzione x: $x\n";
increment($x);
echo "Dopo la funzione x: $x\n";
```

2.6 - Parametri di Default

Abbiamo visto che, al momento della chiamata, una funzione deve ricevere tanti parametri reali quanti sono i formali nella sua dichiarazione.

Possiamo settare dei parametri di default che verranno utilizzati se non riusciamo a passare il giusto numero di parametri.

```
$num1 = 1;


function sum($a, $b = 0)
{
  echo $a + $b;
}

sum($num1);
```

2.7 - Splat Operator - Parametri variabili

Ci sono casi (rarissimi) in cui non sappiamo quanti parametri verranno mandati alla nostra funzione.

```
$a = 1;
$b = 6;
$c = 9;
$d = 11;

function sum(...$numbers)
{
  var_dump($numbers);
  //$numbers = [1,6,3,11];
}

sum($a, $b, $c, $d);
```

2.8 Return Types In PHP

```
function stampaIntero()
{
```

```
    return 'ciao';
}

stampaIntero();
```

2.9 Named Arguments

```
function saluta($nome, $cognome)
{
    echo "$nome $cognome";
}

saluta(cognome: 'Mansi', nome: 'Francesco');
```

3 - Valori di ritorno RETURN

Abbiamo visto fino ad ora che una funzione ci può mostrare il risultato per cui è stata creata.

Ci sono dei casi in cui una funzione deve semplicemente ritornare un valore.

```
$students = [
  [
    'name' ⇒ 'Marco',
    'age' ⇒ 21,
  ],
  [
    'name' ⇒ 'Maria',
    'age' ⇒ 17,
  ],
  [
    'name' ⇒ 'Jack',
    'age' ⇒ 24,
  ],
];
```

```
function avg($array)
{
  $sum = 0;
  foreach ($array as $value) {
    $sum = $sum + $value['age'];
  }

  return $sum / count($array);
}
$media = avg($students);

echo "Età media degli studenti è: " . round($media, 2);
```

Esempio ritorno booleano

```
$students = [
  [
    'name' ⇒ 'Marco',
    'age' ⇒ 21,
  ],
  [
    'name' ⇒ 'Maria',
    'age' ⇒ 17,
  ],
  [
    'name' ⇒ 'Jack',
    'age' ⇒ 24,
  ],
];



function isAdult($var)
{
  if ($var >= 18) {
    return true;
  }
```

```
  return false;
}

foreach ($students  as $value) {
  if (isAdult($value['age'])) {
    echo $value["name"];
  }
}
```

4 - Live Coding - Check Password

1) La password deve essere almeno 8 caratteri

2) Deve contenere almeno un numero

3) Deve contenere almeno un carattere maiuscolo

1) Primo Step - La password deve essere almeno 8 caratteri

```
$password = readline('Inserisci una Password: ');


if(strlen($password) >= 8){
    echo "La Password è valida";
} else {
    echo "La Password non è valida";
}
```

2) Secondo Step - Deve contenere almeno un numero

Cicliamo tutte le lettere della stringa:

```
for ($i = 0; $i < strlen($password); $i++) {
  echo $password[$i] . "\n";
}
```

Come posso cercare se è un numero? Cerco su google:

```
php check if string is numeric
```

```
for ($i = 0; $i < strlen($password); $i++) {
  if (is_numeric($password[$i])) {
    echo "La Password è valida\n";
    break;
  } else {
    echo "La Password non è valida\n";
  }
}
```

3) Terzo Step - Deve contenere almeno un carattere maiuscolo

Cerco su google:

```
how to check uppercase php
```

E scriveremo:

```
for ($i = 0; $i < strlen($password); $i++) {
  if (ctype_upper($password[$i])) {
    echo "La Password è valida\n";
    break;
  } else {
    echo "La Password non è valida\n";
  }
}
```

Il codice è tutto procedurale, è una schifezza! Dobbiamo trasformarlo in Funzionale

1) Primo Step - La password deve essere almeno 8 caratteri

```
$password = readline('Inserisci una Password: ');

function checkLenght($string, $max = 8)
{
  if (strlen($string) >= $max) {
    return true;
  } else {
    return false;
  }
}
```

2) Secondo Step - Deve contenere almeno un numero

```
function checkIsNumeric($string)
{
  for ($i = 0; $i < strlen($string); $i++) {
    if (is_numeric($string[$i])) {
      return true;
    }
  }
  return false;
}
```

3) Terzo Step - Deve contenere almeno un carattere maiuscolo

```
function checkIsUppercase($string)
{
  for ($i = 0; $i < strlen($string); $i++) {
```

```
    if (ctype_upper($string[$i])) {
      return true;
    }
  }
  return false;
}
```

4) Finale

```
$password = readline('Inserisci una Password: ');

$lenght = checkLenght($password);
$numeric = checkIsNumeric($password);
$uppercase = checkIsUppercase($password);

if ($lenght && $numeric && $uppercase) {
  echo "La password è valida";
} else {
  echo "La password non è valida";
}



checkPassword(readline('Inserisci una Password: '));




function checkPassword($string)
{
  if (checkLenght($string) && checkIsNumeric($string) && checkIsUpperca
se($string)) {
    echo "La password è valida";
  } else {
    echo "La password non è valida";
  }
}
```

```
function checkLenght($string, $max = 8)
{
  if (strlen($string) >= $max) {
    return true;
  } else {
    return false;
  }
}

function checkIsNumeric($string)
{
  for ($i = 0; $i < strlen($string); $i++) {
    if (is_numeric($string[$i])) {
      return true;
    }
  }
  return false;
}

function checkIsUppercase($string)
{
  for ($i = 0; $i < strlen($string); $i++) {
    if (ctype_upper($string[$i])) {
      return true;
    }
  }
  return false;
}
```