1. Utilizarea unei funcții

Pentru a folosi o functie, scrieti numele functiei, urmat de o pereche de paranteze. De exemplu, functia ***rand(),*** care genereaza un numar întreg aleator, poate fi apelata astfel:

rand()

Majoritatea functiilor preiau argumente, reprezentând valori, de intrare care influenteaza operarea si rezultatul functiei.

Pentru a specifica argumente, acestea se insereaza între paranteze; daca specificati mai mult de un argument, fiecare argument trebuie separat de vecinul sau printr-o virgula.

Argumentul unei functii poate fi o valoare literala, o variabila sau o expresie.

Unele functii PHP au argumente optionale, care pot fi specificate sau omise, în conformitate cu intentiile dumneavoastra. De exemplu, functia ***rand()*** are doua argumente optionale. Primul argument al functiei indica valoarea întreaga aleatoare cea mai mica pe care o va returna functia; al doilea argument indica valoarea cea mai mare. Daca omiteti ambele argumente, functia returneaza o valoare cuprinsa între 0 si cel mai mare rezultat posibil.

Puteti folosi valoarea returnata de o functie într-o expresie, astfel valoarea va fi accesibila în mod repetat fara a se invoca functia de mai multe ori

Iata un exemplu in care functia ***rand()*** returneaza o valoare aleatoare cuprinsa intre 1 si 500, atribuind valoarea unnei variabile **"$nr"** :

$nr = rand(1, 500);

echo $nr;

Când se produce o eroare în timpul executiei unei functii, PHP genereaza mesaje de eroare. Uneori, asemenea mesaje de eroare sunt nedorite. În acest caz, puteti suprima generarea mesajelor de eroare prin adaugarea in fata numelui functiei a caracterului @.

De exemplu,, pentru a suprima mesajele de eroare care pot aparea în timpul executiei functiei **"f()",** scriem aceasta functie dupa cum urmeaza:

y = @f(x);

Totusi, indicat este sa scrieti scripturi care sa nu genereze erori, ascunderea lor nu rezolva problema ce cauzeaza eroarea.

O functie utila, recomandata a fi folosita in script-uri este functia ***isset()*** si este cel mai des folosita cu ***"if()".***

isset() preia ca argument de obicei o variabila si arata daca aceasta a fost sau nu setata.

De exemplu***: isset($nr)***

- Functia returneaza **TRUE** daca variabila **"$nr"** are setata o valoare (**diferita de NULL**), in caz contrar returneaza **FALSE**.

Aceasta functie este foarte utila in determinarea caror comenzi sa fie executate in functie daca o anumita variabila a fost setata sau nu. Previne aparitia unor erori care apar in cazuri de variabile nule si ajuta si la securitate.

De exemplu, sa presupunem ca avem un cod PHP care vrem sa fie executat numai daca prin adresa URL o fost transmisa o variabila "id", adica o adresa de forma http://www.domeniu.site/script.php?id=un_id, folosim functia isset() astfel:

<?php

```
if (isset($_GET['id'])) {
    // Se executa codul dorit
}
?>
```

- **$_GET['id']** preia valoarea lui **"id"** din URL, iar functia **isset()** verifica daca aceasta valoare exista (daca in URL este "id=ceva"). Daca aceasta exista, returneaza **TRUE** iar functia **"if",** avand astfel valoarea **TRUE**, va executa codul dintre acoladele ei.

Functia **isset()** este utila si pentru situatii de verificarii a inexistentei unei variabile, prin adaugarea caracterului (!) in fata ei. Astfel daca o anumita variabila rezulta a nu fi setata, ii atribuim o valoare sau executam un anume cod special pt. asta.

Exemplu:

```
if (!isset($_GET['id'] {
    die('Pagina apelata este inaccesibila.');
}
```

2. Utilizarea fisierelor incluse

Functiile PHP va permit sa obtineri accesul la programe PHP scrise anterior, create într-un alt fisier extern.

Pentru aceasta puteti folosi functia ***require(),*** care are urmatoarea forma:

require("nume_fisier")

Când este încarcat un script PHP care contine o instructiune ***require,*** continutul fisierului specificat este inserat si executat în script, înlocuind instructiunea require.

De exemplu, sa presupunem ca realizam un site in PHP care este alcatuita din mai multe pagini, iar fiecare pagina contine in partea de sus acelasi cod HTML. In loc sa scriem de fiecare data, pentru fiecare pagina, acelasi cod HTML, il scriem o singura data intr-un fisier **separat (de exemplu "antet.php")** iar in paginile unde vrem sa fie inclus codul HTML folosim functia require()

Prin insertia instructiunii la începutul scriptului din fiecare pagina PHP, ca in exemplu urmator:

```
<?php
require("antet.php");
?>
```

- cu aceasta determinati programul PHP sa includa continutul fisierului **"antet.php"** ca si cum continutul respectiv ar face parte din acel script.

Acest procedeu poate simplifica întretinerea site-ului, deoarece informatiile standard pot fi tinute într-un singur fisier, ceea ce le face usor de localizat si de modificat.

O alta functie, similara instructiunii require este functia **include().** Spre deosebire de functia **require()** ***care introduce datele din fisierul extern intocmai cum sunt scrise***, ***functia include() este o instructiune executabila ce determina evaluarea scriptului PHP din fisierul extern si codul acestuia este executat ca si cum ar fi aparut în textul scriptului unde este inclus.***

Sintaxa functiei include() este urmatoarea:

include("nume_fisier.php");

Functia corelata ***require_once()*** asigura faptul ca fisierul specificat este inclus o singura data într-un script dat. În cazul în care creati fisiere externe care si ele folosesc instructiunea

require pentru a include continutul altor fisiere externe, puteti gasi instructiunea require_once utila.

3. Definirea unei funcții

În afara de a utiliza functiile din biblioteca de functii a limbajului PHP, va puteti defini si folosi propriile functii.

Pentru a defini o functie, in PHP functiile incep intotdeauna cu declaratia: **function**, ca in exemplul urmator:

function nume_functie(nume_argument) {
    bloc de instructiuni
}

Cuvântul cheie **"function", "numele_functiei" si "nume_argument"** alcatuiesc antetul functiei. Termenul de corp al functiei se refera la instructiunile incluse între acolade care urmeaza dupa antetul functiei. Instructiunile din corpul functiei sunt executate atunci când functia este apelata.

Numele functiilor nu prezinta sensibilitate la diferenta între majuscule si minuscule; ca atare, "f()" si "F()" reprezinta referiri la aceeasi functie.

Daca doriti sa definiti o functie care nu are argumente, **puteti omite "nume_argument";** daca doriti sa definiti o functie cu mai multe argumente, puteti include argumente suplimentare dupa "nume_argument", fiecare argument fiind separat de vecinul sau printr-o virgula. Parantezele si numele argumentelor incluse între acestea poarta numele de lista cu argumente.

1)Ca exemplu, iata o **functie care calculeaza aria unui dreptunghi**:

```
<?php
function arie($lungime, $latime) {
    return $lungime * $latime;
}
?>
```

Lista cu argumente a functiei **"arie()"** include argumentele **$latime si $inaltime.** Corpul functiei este alcatuit dintr-o singura instructiune; cu toate acestea, corpul unei functii poate contine mai multe instructiuni.

Daca doriti ca o functie sa returneze o valoare, trebuie sa determinati functia sa execute o instructiune return care furnizeaza valoarea respectiva.

Instructiunea return determina sistarea executarii functiei; nu este necesar ca aceasta sa fie ultima instructiune fizica din corpul functiei. Daca definiti o functie care nu are nici o instructiune return (sau pentru date de iesire, precum "echo"), functia va returna valoarea speciala NULL.

2)Exemplu: suma a trei numere

```
<?php
function suma($a, $b, $c) {
        $s = $a + $b + $c;
        echo 'suma este '.$s;
}
/*
```

```
ceva cod php
*/

suma(5, 10, 30);  //apelarea functiei
?>
```

3) **Exemplu:aria unui cerc**

```
<?php
function aria() {
        define("pi", 3.14);
        $raza = 3;
        $arie = pi * $raza * $raza;
        return $arie;
}
/*
ceva cod php
*/

echo 'aria cercului este '.aria(); //apelarea functiei
?>
```

4. Apelarea unei funcții definite de utilizator

O functie definita de utilizator poate fi apelata ca orice functie.
De exemplu, iata o instructiune care apeleaza functia "arie()" din exemplul anterior:

```
<?php
function arie($lungime, $latime) {
      return $lungime * $latime;
}
$rezultat = arie(5,3);
echo "Aria exte : $rezultat";
?>
```

Valorile argumentelor 5 si 3 le înlocuiesc pe acelea ale argumentelor din corpul functiei, care se comporta ca si cum ar fi fost scrisa astfel:

return 5*3

Rezultatul afisat al acestui script va fi :
Aria exte : 15

5. Terminarea executiei unui script

O instructiune return determina sistarea executiei functiei care o contine. In cazul în care doriti sa sistati prelucrarea unui întreg script, puteti invoca functia **exit().**

Iata un exemplu simplu:

```
<?php
function stop() {
      exit();
}
```

```
echo "<br />Unu...";
echo "<br />Doi...";
stop();
echo "<br />Trei...";
?>
```

Dupa executie, scriptul afiseaza:

Unu...

Doi...

Rezultatul acestui script include cuvintele "Unu" si "Doi", dar nu si cuvântul "Trei". Prin apelarea functiei "stop()" se executa corpul functiei respective; la invocarea functiei, exit(), executia scriptului se încheie.

6. Functii recursive

Este posibil ca o functie din PHP sa se auto-apeleze. O functie care procedeaza astfel se numeste **"functie recursiva".** Totusi, daca nu aveti experienta de programare, este recomandat sa nu scrieti functii recursive. Cu toate acestea, puteti scrie accidental sau intalni o asemenea functie, deci este util sa stiti unele notiuni referitoare la aceasta.

Studiati scriptul urmator care defineste si invoca o functie recursiva simpla:

```
<?php
function recursor($nr) {
    $nr++;
    if ($nr<8) {
        return recursor($nr);
    }

    return $nr;
}

$x = recursor(3);
echo $x;
?>
```

- Daca rulati acest script, rezultatul afisat va fi 8.

- Variabila **"$x"** primeste ca valoare functia "recursor()" careia ii transmite ca argument numarul 3.

- Functia **"recursor()"** incrementeaza cu o unitate valoarea argumentului, prin **"$nr++;"** care devine 4, apoi conditia **"if ($nr<8)"** verifica daca aceasta variabila din functie e mai mica decat 8, in caz afirmativ se executa comanda "return recursor($nr);" care sisteaza executia altui cod din functie si auto-apeleaza iar functia (cu noua valoare a lui "$nr" ca argument) care iar incrementeaza valoarea lui "$nr" si verifica din nou conditia care iar autoapeleaza functia, ... si tot asa pana cand "$nr" va avea valoarea 8 si la verificare conditiei "if" aceasta returneaza FALSE si se trece mai departe la executia comenzii "return $nr;" care va returna 8 ca valoare a variabilei "$x".

Functia "echo" va afisa valoarea lui "$x", adica 8.

7. Definirea argumentelor prestabilite

PHP va permite sa definiti functii cu argumente prestabilite. Daca apelati o functie care are un argument prestabilit, dar nu furnizati nici o valoare pentru argumentul respectiv, argumentul ia o valoare prestabilita specificata la inceput.

Iata un exemplu simplu, studiati-l cu atentie:

```
<?php
function impozit_vanzari($cantitate, $rata = 0.0725) {
    return $cantitate * $rata;
}

$cumparaturi = 123.45;
echo "<br />cumparaturi = $cumparaturi";
$impozit1 = impozit_vanzari($cumparaturi, 0.09);
echo "<br />impozit1 = $impozit1";

$cumparaturi = 123.45;
echo "<br /><br />cumparaturi = $cumparaturi";
$impozit2 = impozit_vanzari($cumparaturi);
echo "<br />impozit2 = $impozit2";
?>
```

Rezultatul afisat va fi:
cumparaturi = 123.45
impozit1 = 11.1105

cumparaturi = 123.45
impozit2 = 8.950125

- **Functia impozit_vanzari** preia doua argumente: un argument obligatoriu, denumit **$cantitate**, si un argument prestabilit, denumit **$rata.**

Daca apelati functia si furnizati un singur argument, valoarea argumentului respectiv se considera ca fiind valoarea argumentului $cantitate, iar valoarea 0.0725 se foloseste ca valoare a argumentului $rata. Astfel, la prima invocare a functiei, pentru "impozit1", $rata are valoarea 0.09, specificata drept al doilea argument al functiei. Cu toate acestea, la a doua invocare a functiei, pentru "impozit2", $rata are valoarea 0.0725 deoarece este specificata valoarea unui singur argument, si astfel "$rata" a preluat valoarea prestabilita initial.