PHP Dili

PHP yorumlayicisi, bu "programi" çalistirabilmek için dosyanin içinde PHP komutlarini arar. PHP komutlari birinci bölümde gördügümüz gibi iki sekilde yazilabilir:

PHP:

```
1.
2.    <?php .... ?>
3.    <? .... ?>
```

Bunlara PHP komut ayraci denir; birinci türü uzun veya standart ayraç sayilir; ikincisine ike "kisa ayraç" denir.

PHP kodlarimiz, olusturulmasini istedigimiz sayfanin HTML kodlari ile tabir yerinde ise içiçe yazilir.

PHP:

```
1.
2.    <?php
3.       print "Merhaba Dünya!";
4.    ?>
```

Sayfalara yorum eklemek için;

PHP:

```
1.
2.  <HTML>
3.  <!- - Bu satir HTML'in yorum satiri
4.  Buraya istedigimiz kadar yorum yazabiliriz..
5.  Browser bu satirlari dikkate almaz - - >
6.  <HEAD>
7.  <TITLE>PHP ile Merhaba</TITLE>
8.  </HEAD>
9.  <BODY>
10. <CENTER>
11. <B>
12. <H1>
13. <?php
14. /*
15. Bu satir da PHP'nin çok-satirli yorum bölümü..
16. Bunu da PHP yorumcusu asla dikkate almaz
17. Buraya istedigimiz kadar yorum yazabiliriz.
18. */
19.     print "Merhaba Dünya!";
20. // Bu ise PHP'nin tek satirli yorum bölümü
21. # Bu satirlari da PHP yorumcusu dikkate almayacaktir.
22. ?>
23. </H1>
24. </B>
25. </CENTER>
26. </BODY>
27. </HTML>
```

Degiskenler

PHP'de de, bir çok baska bilgisayar programlama dilinde oldugu gibi degiskenlerin içine bir deger konmadan önce tanimlanmasi mümkündür; fakat gerekli degildir. Degiskenleri adinin önüne $ isareti koyarak tanimlariz:

PHP:

```
1.
2.  <?php
3.  $adi;
4.  $soyadi;
5.  $123;
6.  $sevdigiRenk;
7.  ?>
```

PHP'de genellikle degiskenleri degerini atayarak belirleriz:

PHP:

```
1.
2.  <?php
3.  $adi = "Resit";
4.  $soyadi = "Gülen";
5.  $123 = 123;
6.  $sevdigiRenk = "yesil";
7.  ?>
```

Degiskenler, kullanildiklari isleme, tasidiklari degeri verirler:

PHP:

```
1.
2.  <?php
3.  print $adi;
4.  ?>
```

PHP'de özel bir degiskene degisken adi olarak kullanilacak degerleri de atayabiliriz:

PHP:

```
1.
2.  <?php
3.  $adi     = "Resit";
4.  $degisken = "adi";
5.  print $$degisken;
6.  ?>
```

Burada Browser penceresine yine "Resit" kelimesi yazilacaktir; çünkü PHP $degisken adli degiskenin "adi" adli degiskeni tuttugunu bilecek ve iki Dolar isaretini görünce, $degisken'in degerini degil, onun tuttugu degiskenin degerini yazacaktir.

Veri Türleri

PHP açisindan dünyada alti tür deger vardir:

Tamsayi	(Integer): 5,124, 9834 gibi
Çift	(Double) : 3,567 gibi
Alfanümerik	(String) : "Resit" gibi
Mantiksal	(Boolean): dogru (true)/yanlis (false) gibi
Nesne	(Object)
Dizi	(Array)

Tür Degistirme

Bir degiskenin degerinin türü hakkinda kuskunuz varsa, en emin yol bunu PHP'nin kendisine sormaktir. Bu sorgulamayi gettype() fonksiyonu ile yapariz.

Simdi, bir PHP programi yazalim, bir takim degiskenlere degerler atayalim ve bunlarin türlerini PHP'ye soralim.

PHP:

```
1.
2.  <?php
3.     $sayi  = 5;
4.     print("Birinci degiskenin adi: \$sayi<br>");
5.     print("Degeri : ");
6.     print "$sayi<br>";
7.     print("Türü : ");
8.     print gettype( $sayi ) ; //tamsayi/integer
9.     print "<br>";
10.    print "<br>";
11.
12.    $alfanumerik  = "Resit";
13.    print "Ikinci degiskenin adi: \$alfanumerik<br>";
14.    print "Degeri : ";
15.    print "$alfanumerik<br>";
16.    print("Türü : ");
17.    print gettype( $alfanumerik ) ; //alfanümerik/string
18.    print "<br>";
19.    print "<br>";
20.
21.    $ondalik  = 5.1234;
22.    print "Üçüncü degiskenin adi: \$ondalik<br>";
23.    print "Degeri : ";
24.    print "$ondalik<br>";
25.    print("Türü : ");
26.    print gettype( $ondalik ) ; //çift,ondalik/double
27.    print "<br>";
28.    print "<br>";
29.
30.    $mantiksal  = true;
31.    print "Üçüncü degiskenin adi: \$mantiksal<br>";
32.    print "Degeri : ";
33.    print "$mantiksal<br>";
34.    print("Türü : ");
35.    print gettype( $mantiksal ) ; //mantiksal/boolean
36.    print "<br>";
37.    print "<br>";
38. ?>
```

Burada mantiksal (boolean) deger olarak dogru anlamina true degeri atadigimiz halde, PHP'nin bu degiskenin degeri olarak 1'i gösterdigine dikkat edin. PHP'de bir fonksiyon, elde ettigi deger dogru ise sonuç olarak 1 degerini verir.

Escape

Su satirdaki, ters-bölü isareti dikkatinizden kaçmamis olmali:

PHP:

```
1.
2.  <?php
3.      print "Ikinci degiskenin adi: \$alfanumerik<br>";
4.  ?>
```

PHP için özel anlami olan isaretlerin anlamlandirilmasini önlemek ve bu isaretleri düz metin saymasini saglamak için bu isaretlerin önüne ters-bölü isareti koyariz.
söyledir:

\'	Tek tirnak
\"	Çift tirnak
\\	Ters-bölü
\$	Dolar isareti
\n	Yeni Satir (New Line)
\r	Satir Basi (Return)
\t	Sekme (Tab) karakteri

Kimi zaman bir degiskene atadigimiz degerin türünü degistirmek gerekir. Bunu settype() fonksiyonu ile yapariz.

PHP:

```
1.
2.   <?php
3.   $degisken  = 5.67890;
4.      print("Degiskenin  degeri : ");
5.      print "$degisken<br>";
6.      print("Türü : ");
7.      print gettype( $degisken ) ; //çift,ondalik/double
8.      print "<br>";
9.      print "<br>";
10.
11.     print "Ilk degistirme islemi: Alfanümerik/String:<br>";
12.     settype( $degisken, string ); //alfanümerik/string (integer,double,booleon)
13.     print "Degeri : ";
14.     print "$degisken<br>";
15.     print("Türü : ");
16.     print gettype( $degisken ) ; //alfanümerik/string
17.     print "<br>";
18.     print "<br>";
19.  ?>
```

Dört yararli fonksiyon

isset() ve unset()

isset() fnksiyonu, PHP'nin bir degiskenin içinde deger bulunup bulunmadigini sinamasini saglar. unset() ise varolan bir degiskeni yok eder.

```
if (isset($bir_degisken)) {
print( $bir_degisken );
}
else {
unset($bir_degisken);
}
```

Bu kod parçasi, $bir_degisken isimli dgiskenin içi bos degisse, içerigini görüntüleyecek, içi bos ise varligina son verecektir.

empty()

isset() fonksiyonun tersi isleve sahiptir; bir degiskene deger atanmamissa, veya degeri sifir veya bos alfanümerik (null string) ise, dogru (True) degeri verir.

```php
<?php
   $a = 6.567;
if (is_double($a)) {
   print ("A Double'dir<br>");
   }
   $b = "Resit";
if (is_double($a)) {
   print ("B String'dir<br>");
   }
   $c = 6;
if (is_int($c)) {
   print ("C Integer'dir<br>");
   }
?>
```

```php
<?php
$sayi=4.5;
echo floor($sayi);      =>> 4 (Sayiyi asagiya yuvarlar)
echo ceil($sayi);       =>> 5 (Sayiyi yukariya yuvarlar)
echo round($sayi);      =>> 5 (Sayiyi yuvarlar)

echo max(13,12,5,7);    =>> 13
echo min(13,12,5,7);    =>> 5
?>
```

Bu kod, Browser penceresine "A double'dir, B String'dir, C Integer'dir" yazdiracaktir. PHP'de bu fonksiyonlara benzeyen fakat baska tür deger arayan su fonksiyonlar da vardir: is_array(), is_object.

Islemciler (Operatörler)

Aritmetik islemciler:

+	Toplama		6+5	=	11
-	Çikartma	6-5	=	1	
/	Bölme		6/5	=	1.2
*	Çarpma		6*5	=	30
%	Kalan (Modulus)	6%5	=	1	

Rastgele Sayi

PHP:

```
1.
2.   <?php
3.   srand((double) microtime()*1000000);
4.   echo rand(20,30)       =>> (20-30 arasi sayi döndürür)
5.   ?>
```

PHP'nin atama islemcisinin esittir (=) isareti oldugunu hatirliyorsunuz birlesik-atama (combined-assignment) islemcileri, bu isarete diger aritmetik islemciler eklenerek olusturulur.

Islemci	Örnek	Anlami
+=	$a += 5	$a = $a + 5
-=	$a -= 5	$a = $a - 5
/=	$a /=5	$a = $a / 5
*=	$a *= 5	$a = $a * 5
%=	$a %= 5	$a = $a % 5
.=	$a .= "metin"	$a = $a" metin"

Bir Arttirmak veya Azaltmak için

Degerleri sadece 1 arttirmak veya azaltmak için PHP, bir kolaylik saglar:

$a++ veya ++$a : $a'nin degerini 1 arttirir;
$a-- veya --$a : $a'nin degerini 1 eksiltir.

PHP'nin karsilastirma yapmasi için kullandigimiz islemciler ise islem isaretinin sagi ve solundaki degerleri veya degiskenlerin degerlerini isaretin belirttigi karsilastirmayi yaptiktan sonra ortaya ya dogru (true) ya da yanlis (false) sonucunu çikartirlar.

Islemci	Örnek	Örnek	$a=6 ise:
==	esitse	$a == 5	Yanlis/False
!=	esit degilse	$a != 5	Dogru/True
===	ayni ise	$a === 5	Yanlis/False
>	büyükse	$a > 5	Doguru/True
<	küçükse	$a < 5	Yanlis/False
<=	küçükse veya esitse	$a <= 5	Yanlis/False
>=	büyükse veya esitse	$a >= 5	Dogru/True

PHP'de bu karsilastirmayi iki grubun arasina koydugumuz isaretlerle yapariz Isaretin sag ve sol tarafinin dogrulugu veya yanlisigi isarete göre nihai sonucun dogru veya yanlis olmasini saglar. Bu karsilastirmalari yaparken su islemcileri kullaniriz:

Islemci	Adi	Anlami		Örnek
||	veya	sol veya sag dogru		dogru || yanlis = dogru
or	veya	sol veya sag dogru		dogru || yanlis = dogru
&&	ve	sol ve sag dogru		dogru || yanlis = yanlis
and	ve	sol ve sag dogru		dogru || yanlis = yanlis
Xor		Sartli-veya	Sadece sol veya sag	dogru dogru || yanlis=dogru
!	Degil	sol veya sag yanlis		dogru || yanlis = dogru

PHP:

```
1.
2.  <?php
3.     $vize = 45;
4.     $final = 65;
5.     if ($vize >= 50 && $final >= 50) {
6.     print ("Ögrenci geçti!");
7.     }
8.     else {
9.     print ("Ögrenci kaldi!");
10.    }
11. ?>
```

Sabit Degerler

define ("SABIT_DEGER", deger);

Burada SABIT_DEGER yerine, tanimlamak istedigimiz sabit degere verecegimiz isim, deger yerine de sabit degeri yazariz. Örnek:

PHP:

```
1.
2.  <?php
3.     $Dolar_miktar = 125;
4.     define ( "DOLAR_KURU", 625675);
5.     $TL_Tutar = $Dolar_miktar * DOLAR_KURU;
6.     print ($TL_Tutar);
7.  ?>
```

Tanimlanmis olan bir sabiti yeniden olusturamayiz; ama buna tesebbüs ettigimizde PHP hata vermez. Bir sabit degerin olusturulmus olup olmadigini defined() fonksiyonu ile anlayabiliriz:

Tanimlanmis olan bir sabiti yeniden olusturamayiz; ama buna tesebbüs ettigimizde PHP hata vermez. Bir sabit degerin olusturulmus olup olmadigini defined() fonksiyonu ile anlayabiliriz:

PHP:

```
1.
2.   <?php
3.      $Dolar_miktar = 125;
4.      if (defined( "DOLAR_KURU" )) {
5.      echo ("Sabit deger daha önce tanimlanmisti.<br>");
6.      }
7.   ?>
```

Dizi-Degiskenler

Dizi degiskenden ayrintili söz edebilmek için önce tipik bir dizi-degiskende neler olduguna bakalim. Sözgelimi, verdiginiz "PHP ile Programlama" kursundaki ögrencilerinizin listesi söyle olabilir:

Dizi Degisken Olusturalim
Simdi, PHP bize öyle bir araç vermeli ki, biz bir kerede bu listenin tümünü, her bir ögesine sanki bir degiskenin degeri imis gibi tek-tek, veya bir kaçina birden ulasabilmeli ve arzu ettigimiz zaman notlari doldurabilmeliyiz. Ögrenciler de yapacagimiz Web sitesine girerek, kendi notlarini görebilmeli ve notlarini inceleyebilmeli. PHP'nin bu amaçla sagladigi araç, çok-boyutlu dizi-degisken olusturma aracidir. Ve bu araçla yukaridaki listeyi aynen söyle yapabiliriz.

```
1.
2.   <?php
3.   $ogrenciler = array (
4.   array ( adi => "Özbay", soyadi => "Altun", sinav1 =>"", sinav2 =>"", not =>""),
5.   array ( adi => "Muharrem", soyadi => "Taç", sinav1 =>"", sinav2 =>"", not =>""),
6.   array ( adi => "Hasan", soyadi => "Civelek", sinav1 =>"", sinav2 =>"", not =>""),
7.   array ( adi => "Sahika", soyadi => "Tabak", sinav1 =>"", sinav2 =>"", not =>""),
8.   );
9.   // Buraya baska kodlar girecek
10.  print $ogrenciler[0][adi];
11.  ?>
```

Programdaki "print()" komutunu sadece dizi degiskeni dogru yazip yazmadigimizi sinamak amaciyla yazdik; bu programi Browser'da açtiginizda yazdiginiz ilk ismi Browser penceresinde görüyorsaniz, dizi-degiskeni dogru sekilde olusturdunuz demektir. Burada, array() komutnu yazarken, süslü parantez degil, normal parantez kullandigimiza ve herbir elemanin degerlerinin sonunda vrigül olduguna dikkat edir. Bir diger önemli nokta: endeks adlari bir kelimeden fazla ise bunlari tirnak içine alarak belirtmektir. Örnegin:

array (adi => "Özbay", soyadi => "Altun", "Sinav 1 Notlari" =>"", "Sinav 2 Notlari" =>"", "Toplam Not Ortalamasi" =>""),

Burada, daha öncekilere benzer bir sekilde adlandirilmis $ogrenciler degiskenin içerigini array() komutu ile doldurdugumuzu görüyoruz. Array() ile böyle çok boyutlu ve içerdigi degerlerin her birinin bir "endeks adi" olan dizi-degiskene Iliskili Dizi (Associative array) de denir. Perl bilenler ise bu tür degiskenlere "Hash" dendigini hatirlayacaklardir. Iliskili Dizi'lerin birinci satiri 0, ikinci satiri 1, üçüncü satiri 2.. diye numaralandirilir. Bu dizinin o satirindaki kaydin sira endeksidir. Ayrica burada "adi," "soyadi," "sinav1" .. dizi degiskenin içindeki degerlerin endeks adidir. Yani bu degerlere atifta bulunurken, referans yaparken veya bu degerleriekullanmak amaciyla erisirken sira endeksi ve endeks adiyla hitabederiz. Yukaridaki sinama amaçli print() komutuna bakarsaniz, birinci ögrencinin ismini "[0][adi]" olarak çagiriyor.

Çok elemanli iliskili dizi olusturmanin bir diger yolu, yeri geldiginde böyle bir dizi için yeni bir üye ilgili bilgileri eleman endeksi ve degerler için endeks adi belirterek söyle bir kod yazmaktan ibarettir.

PHP:

```
1.
2.  <?php
3.      $ogrenciler[0][adi] = "Özbay";
4.      $ogrenciler[0][soyadi] = "Altun";
5.      $ogrenciler[0][sinav1] = "";
6.      $ogrenciler[0][sinav2] = "";
7.      $ogrenciler[0][not] = "";
8.      // Buraya Buraya baska kodlar girecek
9.      print $ogrenciler[0][adi];
10.
11. ?>
```

Bir dizi degiskende kaç boyut olacaksa, o kadar içiçe array() ögesi olusturabiliriz. Buna göre tek boyutlu bir dizi degisken sadece bir array() komutu ile ve sadece degerler verilerek olusturulabilir. Diyelim ki yukaridaki ögrenci listemiz sadece ögrencilerin isimlerinden olusacak. Bu durumda $ogrenciler degiskenine iliskin satiri söyle yazabilirdik:

$ogrenciler = array ("Özbay", "Muharrem", "Hasan", "Sahika");

PHP, böyle tek boyutlu bir dizinin örnegin birinci elemanini, "$ogrenciler[0]" adiyla bilir. Böyle bir tek-boyutlu diziyi olusturmak için PHP bize baska bir kolaylik da saglar: array() komutunu kullanmadan, dogruca dizinin ögelerine deger vermemiz mümkündür. Yukaridaki programin sadece PHP bölümünü söyle degistirerek kaydedin:

PHP:

```
1.
2.  <?php
3.        $ogrenciler[] = "Özbay";
4.        $ogrenciler[] = "Muharrem";
5.        $ogrenciler[] = "Hasan";
6.        $ogrenciler[] = "Sahika";
7.  // Buraya baska kodlar girecek
8.     print $ogrenciler[0];
9.  ?>
```

Böyle sirayla dizi degisken olusturur veya olusturulmus bir dizi degiskene ek yaparken, degiskenin sira numarasini yazmazsak, PHP bunlari kendisi siralar. Yukaridaki kodun da Browser penceresine "Özbay" yazdirmasi gerekir. Mevcut tek-boyutlu bir dizi degiskene ek yaptigimizda, be yeni degerin dizinin en altina eklenmesini istiyorsak, sira numarasi yazmamiza gerek yoktur. Mevcut degerlerden birini degistirmek istiyorsak, o degerin sira numarasini yazmamiz gerekir. Bunu denemek için yukaridaki kodu söyle degistirilim

```
1.
2.   <?php
3.         $ogrenciler[] = "Özbay";
4.         $ogrenciler[] = "Muharrem";
5.         $ogrenciler[] = "Hasan";
6.         $ogrenciler[] = "Sahika";
7.         // Buraya baska kodlar girecek
8.         $ogrenciler[0] = "Emre";
9.         $ogrenciler[15] = "Özbay";
10.
11.      print ("Dizideki 1'nci isim: $ogrenciler[0] <br>");
12.      print ("Dizideki 2'nci isim: $ogrenciler[1] <br>");
13.      print ("Dizideki 3'ünci isim: $ogrenciler[2] <br>");
14.      print ("Dizideki 4'üncü isim: $ogrenciler[3] <br>");
15.      print ("Dizideki 5'inci isim: $ogrenciler[4] <br>");
16.      print ("Dizideki 6'nci isim: $ogrenciler[5] <br>");
17.      print ("..............<br>");
18.      print ("Dizideki 15'nci isim: $ogrenciler[15] <br>");
19.
20.  ?>
```

```
1.
2.   <?php
3.      $ogrenci[adi] = "Özbay";
4.      $ogrenci[soyadi] = "Altun";
5.      $ogrenci[sinav1] = "";
6.      $ogrenci[sinav2] = "";
7.      $ogrenci[not] = "";
8.      // Buraya baska kodlar girecek
9.      print $ogrenci[adi];
10.  ?>
```

Bu programin Browser penceresine gönderecegi sirada, birinci ögrenci ($ogrenci[0]) olarak bu kez Özbay degil Emre yazdigini görecegiz.

Bunun sebebi, diziyi olusturan ilk grup deyimden sonra,

$ogrenciler[0] = "Emre";

satiri ile birinci elemanin degerini degistirmis olduk. 15'nci elemana atama yapmakla, PHP'nin $ogrenciler dizisinde 6, 7, 8, 9,.. 14'e kadar bos elemanlar olusturmasina sebep olduk.
Tek boyutlu dizileri de Iliskili Dizi olarak olusturabilir yani degerlere endeks adi verebiliriz.

PHP, $ogrenci adli degiskenin bes ayri degeri oldugunu ve bunlarin "adi," "soyadi," "sinav1"... oldugunu biliyor. Simdi artik istedigimiz noktada bu degiskenin istedigimiz degerine, o degerin endeks adini yazarak, çagrida bulunabiliriz; bu degeri yeniden verebiliriz.

Dizi degiskenleri kullanalim
Yukaridaki paragrafta "..degiskenin istedigimiz degerine, o degerin endeks adini yazarak, çagrida bulunabiliriz.." dedigimizi görmüs olmalisiniz. Dizi veya tekil, degiskenleri olusturmamizin sebebi, tuttuklari degerleri programimizin geregi olan sekilde ve yerde kullanmaktir. Sadece bir deger tutan degiskenleri örnegin print() komutu ile sik sik kullandik. Yukarida dizi degisken örneklerinde de bazi degiskenleri ve degerlerini çagirdik. Ancak dizi degiskenlerin degerlerinden yararlanabilmek için baska araçlar da vardir.

Herseyden önce dizi degiskenlerin büyüklügü, boyutu bizim için önem tasiyabilir. Özellikle bir veritabani dosyasini okutarak olusturacagimiz dizi degiskenin kaç elemani ve her bir elemanin kaç ögesi bulundugunu bilmemiz gerekebilir.

Bir dizi degiskenin kaç elemani bulundugu, o degiskenin count() özelligi sorgulanarak ögrenilir. count(), dizideki eleman sayisini verir. Simdi bunu bir örnekle görelim.

```
1.
2.  <?php
3.          $ogrenciler[] = "Özbay";
4.          $ogrenciler[] = "Muharrem";
5.          $ogrenciler[] = "Hasan";
6.          $ogrenciler[] = "Sahika";
7.          // Buraya baska kodlar girecek
8.  print ("\$ogrenciler adli dizide ". count($ogrenciler) ." adet eleman var.");
9.  ?>
```

Bu program Browser penceresine dizimizde 4 eleman bulundugunu bildirecektir. Simdi isleri biraz karmasik hale getirelim! Yukaridaki kodun, print() satirinin yerine su satirlari ekleyerek, kaydedelim.

PHP:

```
1.
2.   <?php
3.   print ("\$ogrenciler adli dizide ". count($ogrenciler) ." adet eleman var.");
4.   print ("<br><br>");
5.   for ($sayac=1 ;  $sayac <= count($ogrenciler) ; $sayac++ ) {
6.   print ("\$ogrenciler dizisinin ". $sayac ."'nci elemani: " . $ogrenciler[$sayac] ."<br>");
7.   }
8.   ?>
```

Bu programi çalistirmadan önce, ekledigimiz satirlari irdeleyelim. Ilk print() komutunun Browser penceresine "yazdiracagi" metinde geçen ters bölü isaretini hatirliyor olmalisiniz. Bu, tek veya çift tirnak içine de almis bile olsak, PHP'nin, bir degisken adini gördügü zaman onun yerine o degiskenin tuttugu degeri yazmasi sebebiyle, $ isareti gibi PHP için özel anlami olan isaretlerin anlamlandirilmasini önlemek için yaptigimiz ve adina o karakteri kurtarma veya ESCaping dedigimiz islemdir. Bu islemle, PHP'nin anlamli isaret degil de metin saymasini istedigimiz karakterlerin önüne ters bölü isareti koyariz: \" gibi. Buradaki örnekte, bu sayede PHP "$ogrenciler" kelimesini degisken adi olarak degil, düz metin olarak görüyor. Ki, ayni komutta ayni kelimeyi tekrar kullandigimizda bu kez degisken adi olarak kullaniyoruz ve bu degiskenin count() ögesinin degerini ögreniyoruz. $ogrenci degiskenin "Özbay," "Muharrem," "Hasan" ve "Sahika" degerleri bulunduguna göre, bu degiskenin count()'u 4 olacaktir. ("Ozbay" = 0, .. "Sahika" = 3 olmak üzere..) Bu print() komutu, Browser penceresine tahmin ettiginiz gibi "$ogrenciler adli dizide 4 adet eleman var." yazdiracaktir. Ikinci print() satiri ise ekrana ardarda iki yeni satir isareti gönderecektir.

Simdi karisik noktaya geliyoruz! Burada bir for döngüsü basliyor. Önce döngünün kaç kez tekrar edecegini belirleyecek olan degiskeni tanimliyoruz: $sayac. Sonra bu sayacin kaça kadar çikacagini belirliyoruz. Bu sayiyi, bize yine count() veriyor. Ve tabiî for döngüsünün devam edebilmesi için gerekli son unsur olan, sayacin arttirilmasini saglayan deyim var. Programimiz bu döngünün içinde, yani dört kez, her seferinde dizinin bir elemanin adini Browser penceresine gönderiyor. Simdi, hatirlayacaksiniz, dizi degiskenlerin elemanlarinin bir sira sayisi vardi. Örnegin "Sahika" degeri, dizinin 3 numarali, yani dördüncü elemani; ve bu elemanin degerini ekrana göndermek için su komutu vermemiz yeterli:

```
print ($ogrenciler[4]);
```

Programda ise buradaki endeks sayisini, $sayac degiskeninin o andaki degerinden aliyoruz. Döngünün her seferinde bu deger bir artacagi için bize $ogrenciler degiskeninin o anda hangi elemaninin degeri çagirmak istiyorsak, o elemanin endeksini vermis olacaktir. Ve sonuç olarak programimiz, dizideki bütün degerleri Browser'a gönderecektir.

Kimi zaman buradaki örnekte oldugu gibi, dizinin bütün elemanlarini bir for döngüsüyle degil, foreach döngüsüyle bulmak daha kolay olabilir. Kisaca belirtmek gerekirse, foreach döngüsü, bir dizi degiskenin bütün elemanlari için, arzu ettiginiz isi yapar. foreach döngüsünü yazarken komutun kaç kere icra edilecegini bir sayaçla tutmak gerekmez; çünkü döngü, ona adini verdiginiz degiskenin içindeki bütün degerler bitinceye kadar devam edecektir.

```
1.
2.   <?php
3.   foreach ($ogrenciler as $ogrenci) {
4.       print ("$ogrenci<br>");
5.       }
6.   ?>
```

foreach döngüsü, bir dizi degiskenin adini içinden deger çekilecek kaynak olarak ister; bunu "as" (olarak) kelimesi izler; sonra diziden alinacak her bir degeri geçici olarak tutacak degiskenin adi verilir. Buradaki print() komutumuz, bu geçici degiskenin tuttugu degeri Browser'a gönöderecektir. Bu deger ise döngünün her adiminda dizi degiskendeki bir deger yani ögrencilerin listesi olacaktir.

Dizi elemanlarinin farkli özelliklerine iliskin degerlere endeks adi verdigimiz iliskili dizilerde ise eleman degerlerini çagirmak foreach döngüsünün biraz farkli yazilmasini gerektirir. Perl'e asina alanlarin bu dizi türüne "hash" dendigini hatirlayacaklardir. PHP'de de Perl'ün hash türü degiskenlerinde oldugu gibi, endeks adlarina "anahtar" (key), bu endeksin belirledigi degere ise (evet, dogru tahmin ettiniz!) deger (value) denir. Iliskili dizilerden deger almak üzere foreach döngüsü yazilirken, degerin anahtarini ve degerin kendisini iki geçici degiskene yazmamiz gerekir

```
1.
2.   <?php
3.   foreach ($ogrenciler as $anahtar=>$deger) {
4.   print ("$anahtar = $deger<br>");
5.   }
6.   ?>
```

Bu kodu çalistirmadan önce foreach döngüsü üzerinde kisaca duralim: döngü, $ogrenciler dizisini okumaya basladiginda içinde, benzetme yerinde ise, iki sütun, ve bir çok satirlar bulacaktir. Bu sütunlardan birincisi, ikinci sütundaki verinin adidir; foreach, birinci sütundaki veriyi alarak $anahtar adli geçici degiskenin degeri olarak atayacak; sonra ikinci sütuna geçecek ve bunu alarak $deger adli geçici degiskenin degeri yapacaktir. Döngü, daha sonra print() komutunu icra edecektir. print() ise ve geçici $anahtar degiskeninin degerini, ardindan esittir isaretini ve son olarak da geçici $deger degiskeninin degerini Browser'a gönderecektir. print() komutunun icrasi bitince, foreach, kendisine verdigimiz $ogrenciler degiskeninde anahtar-deger çiftini ele almadigi satir kalip kalmadigina bakacak, ve elemanlarin tümü bitinceye kadar bu islemi tekrar edecektir. Tabiî, sonuç anahtar ve degerlerin altalta siralanmasi olacaktir.

Bir de bu bölümün en basinda ele aldigimiz çok elemanli iliskili diziler vardi. Onlarin içindeki degerleri acaba nasil alabilir ve kullanabiliriz? Tabiî yine bir döngü ile. Fakat bu kez, döngü-içinde-döngü kullanmak zorundayiz. Böyle bir diziyi gözümüzde canlandirirsak, belki neden iki döngüye ihtiyaç oldugununu daha iyi görebiliriz. Gözümüzün önüne bir tablo getirelim: dizinin her bir elemani (bizim ögrenimizde ögrenciler9 bir satirda yer almis olsun; sütunlar olarak da bu elemana ait degerler yer aliyor. Sütun basligi ise, bu degerin endeksi olan anahtar

```
1.
2.   <?php
3.   foreach ( $ogrenciler as $ogrenci ) {
4.      foreach ( $ogrenci as $anahtar => $deger ) {
5.         print ("$anahtar = $deger <br> ");
6.         }
7.      print ("<br>");
8.      }
9.   ?>
```

Kisaca irdelersek, bu kodda foreach döngüsünün önce çok-boyutlu degiskenimizin bir satirini içindeki bütün anahtar+deger çeftleri ile ele alip, tümünü $ogrenci adli degiskene geçici olarak yerlestirdigini görüyoruz. Bu foreach döngüsünün ilk isi yeni bir foreach döngüsü baslatmak oluyor. Yeni foreach ise sazi eline alir almaz, önce, kendisi çok ögeli bir degisken olan (çünkü içinde bir ögrenciye ait, tüm degiskenler ve onlarin endeks adlari var) $ogrenci degiskeninin içindeki anahtar ve deger çiftlerini tek-tek, $anahtar ve $deger degiskenlerine yerlestiriyor; sonra print() komutu ile, aralarina esittir isareti koyarak bu degiskenlerin degerlerini Browser penceresine gönderiyor. Bu döngü biter bitmez, ilk foreach yaptiracagi islere kaldigi yerden devam ediyor; ve ekrana bir yeni satir komutu gönderierek, basa dönüyor; bu kez çok boyutlu dizi degiskenin yeni bir elemana geçiyor. Taa ki, dizinin bütün elemanlari ve elemanlarin bütün ögeleri bitinceye kadar.

Bu noktada bir uyari: Gerçek programda bir dizinin elemanlarina ilk ulastigimizda, elemanin içinde deger bulunup bulunmadigini anlamak yerinde olur. Bunu is_array() fonksiyonu ile yapabiliriz. Bu fonksiyon, dizinin içinde deger varsa, True/Dogru, yoksa False/Yanlis karsiligini verecektir. Buradaki örnekte, ilk foreach satirindan hemen sonra:

```
is_array( $ogrenci )
```

satirini koyarak, dizinin o anda okunan elemanin içinde deger bulunup bulunmadigini anlayabiliriz.

Dizi Degiskenlerin Düzenlenmesi

Dizi degiskenlerin daha verimli sekilde kullanilmasi için PHP bize bir takim araçlar saglar. Bunlarla dizi degiskenleri birlestirebiliriz; içinden kesit alabiliriz, siralayabiliriz veya bazi elemanlarini silebiliriz. Simdi kisaca bu islemleri ele alalim:

Dizileri birlestirme: array_merge()

Iki veya daha fazla dizinin bütün elemanlarini birlestirerek, ortaya yeni bir dizi çikartir.

Örnek:

PHP:

```
1.
2.   <?php
3.   $birinci_dizi = array ( "Özbay" , "Muharrem" , "Hasan" , "Sahika" );
4.   $ikinci_dizi = array ( "Altun" , "Taç" , "Civelek" , "Tabak" );
5.   $yeni_dizi = array_merge ( $birinci_dizi, $ikinci_dizi );
6.   ?>
```

Bu kod ile olusturulan $yeni_dizi isimli dizi degiskenin hangi elemanlara sahip oldugunu, söyle bir kodla görebilirsiniz:

PHP:

```
1.
2.   foreach ( $yeni_dizi as $yeni_eleman ) {
3.       print (" $yeni_eleman <br>");
4.       }
```

Ikinci dizinin bütün elemanlari, birinci dizinin elemanlarinin arkasina eklenmis olmali. array_merge() islemi, çok-boyutlu iliskili dizilere de uygulanabilir; PHP iki dizideki uyumlu-uyumsuz, yani birinde olan digerinde olmayan bütün anahtar+deger çiftlerini yeni dizide de olustur. (array_merge() isleminden sonra birlestirilen dizilerin degismeden kaldigina dikkat edin.)

Dizilere degisken ekleme: array_push()

Bir diziye yeni degiskenler eklemek için, array_push() fonksiyonuna mevcut dizinin adini ve yeni degerleri yazariz. Örnek:

```
$birinci_dizi = array ( "Özbay" , "Muharrem" , "Hasan" , "Sahika" );
$yeni         = array_push ($birinci_dizi, "Altun" , "Taç" , "Civelek" , "Tabak" );
```

Burada $yeni adli degisken sadece $birinci_dizi adli dizinin yeni eleman sayisini tutar. array_push(), kendisine adini verdigimiz dizinin içerigini degistirir. Yukaridaki örnekte içine yeni degerler yazilan dizinin elemanlarini görüntülemek için söyle bir kod yazabiliriz:

```
1.
2.   <?php
3.   print ("\$birinci_dizi adli dizide $yeni_dizi adet degisken var<br>");
4.   foreach ( $birinci_dizi as $ogrenci ) {
5.      print ("$ogrenci <br> ");
6.      }
7.   ?>
```

Dizinin ilk elemanini silme: array_shift()

Bir dizi-degiskenin ilk elemanini tümüyle silmek için array_shift() fonksiyonunu kullaniriz. Bu fonksiyona sadece birinci elemani silinecek dizinin adini vermek yeter.
Örnek:

```
$birinci_dizi = array ( "Özbay" , "Muharrem" , "Hasan" , "Sahika" );
$silinen = array_shift ($birinci_dizi);
```

array_shift(), adini verdiginiz dizinin içerigini degistirir; buradaki örnekte, $silinen adli degisken dizinin silinen birinci elemanin degerini tutar.

Diziden kesit alma: array_slice()

Bir dizi-degiskenin bütün elemanlari yerine bir kesitini kullanmak istiyorsak, bunu array_slice() fonksiyonu ile yapabiliriz. Bu fonksiyona kesit alinacak dizinin adi, kesitin basladigi yer ve kaç adet degisken alinacagi argüman olarak verilir.

Örnek

```
$birinci_dizi = array ( "Özbay" , "Muharrem" , "Hasan" , "Sahika", "Altun" , "Taç" , "Civelek" , "Tabak");
$kesit = array_slice ($birinci_dizi ,  3, 4);
```

Burada, PHP'ye $kesit adli yeni dizi degiskene, $birinci_dizi adli dizinin 3'ncü degerinden itibaren (3 dahil) dört degeri yerlestirmesini bildiriyoruz. array_slice(), adini verdigimiz degiskenin içerigine dokunmaz; yeni dizi degisken olusturulur.

Dizileri siralama: sort() ve rsort()

Bir dizinin içindeki degerleri alfabetik veya küçükten büyüge dogru siralamak için sort() fonksiyonunu kullaniriz.

Örnek:

```
$birinci_dizi = array ( "Özbay" , "Muharrem" , "Hasan" , "Sahika", "Altun" , "Taç" , "Civelek" , "Tabak");
sort ($birinci_dizi);
```

PHP, dizideki bütün degerleri A'dan Z'ye siraya sokacaktir. sort() fonksiyonu dizinin içegini degistirir. Bir diziyi Z'den A'ya veya büyükten küçüge dogru siralamak için de rsort() fonksiyonunu kullanabilirsiniz. (PHP4.0 Türkçe karakterleri tanimiyor.) Bir noktada dikkatli olmak gerekir: bu fonksiyonu iliskili (degerlerin anahtari olarak endeks adi bulunan) dizide kullanirsaniz, PHP, anahtar degerlerini (endeks adlarini) atar, yerine 0'dan itibaren rakam koyar. Bunu önlemek için, iliskili dizileri asort() veya ksort() fonksiyonu ile siralamak gerekir.

Iliskili dizileri siralama: asort() ve ksort()

Iliskili dizilerin diger dizi degiskenlere göre farki, degerlerinin bir de adi bulunmasidir. Degerlerin adlarina anahtar denir. Bir iliskili diziyi degerlerine göre siralamak için asort() fonksiyonu kullanilir.

Örnek:

```
$birinci_dizi = array ( ogr_01=>"Özbay", ogr_02=>"Muharrem" , ogr_013>"Hasan" ,
ogr_04=>"Sahika");
asort ($birinci_dizi);
```

PHP, bu diziyi degerler itibariyle alfabetik siraya sokacaktir. Eger siranin degere göre degil de degerlerin anahtarina (burada ogr_01, ogr_02 olan kelimeler) göre yapilmasini istiyorsak, ksort() fonksiyonunu kullaniriz.

Örnek:

```
$birinci_dizi = array ( ogr_01=>"Özbay", ogr_02=>"Muharrem" , ogr_013>"Hasan" ,
ogr_04=>"Sahika");
ksort ($birinci_dizi);
```

PHP, simdi bu diziyi anahtarlara göre alfabetik siraya sokacaktir.

Metin Düzenleme ve Düzenli Ifadeler

PHP:

```
1.
2.  <?php
3.  substr($degisken,8);
4.  substr ($degisken, 8, 20);
5.  substr($degisken, -9);
6.  trim ($degisken);
7.  strlen($degisken);
8.  strip_tags($metin);          =>> (Metin içersindeki html ve php kodlarini atar)
9.  strtolower($metin);          =>> (Küçük harfe çevirir)
10. strtoupper($metin);          =>> (Büyük harfe çevirir)
11. ucwords($metin);           =>> (Sadece Bas Harfleri büyük yapar)
12. ucfirst($metin);         =>>(Sadece cümlenin bas harflerini büyük yapar)
13. substr($metin,3,5);          =>>(3. karekterden itibaren 5 karekter alir)
14. strpos($metin,'@');          =>>(Istedigim karekterin yerini söyler)
15. strstr($metin,'@');         =>>(Istedigim karekterden sonraki karekterleri alir)
16. substr_count($metin,'@');    =>> (Istedigim karekterden kaçtane oldugunu yazar)
17.
18. $bolumler=explode(',',$metin)    =>> ( (,) ler arasindaki ifadeleri dizi degiskenine aktarir)
19. implode(',',$bolumlar)          =>> ( Dizideki degerleri (,) isareti ile baglar)
20. ?>
```

printf() ve sprintf()

Bu fonksiyonlari bir degiskeni biçimlendirmekte kullaniriz. Birincisinin elde ettigi sonuç ziyaretçinin Browser penceresine gönderilir; ikincisin elde ettigi sonuç ise deger olarak döner. Önce bu fonksiyonlarla kullanabilecegimiz biçim parametrelerini siralayalim:

%	Yüzde isareti. Yaninda biçim parametresi gerekmez.
b	Degisken tamsayi olarak islem görür ve ikili sayi olarak döner.
c	Degisken tamsayi olarak islem görür ve ASCII degerinin karsiligi olan karakter olarak döner.
d	Degisken tamsayi olarak islem görür ve ondalik sayi olarak döner.
f	Degisken kesirli sayi olarak islem görür ve kesirli sayi olarak döner.
o	Degisken tamsayi olarak islem görür ve sekiz-tabanli (octal) sayi olarak döner.
s	Degisken alfanümerik olarak islem görür ve alfanümerik olarak döner.
x	Degisken tamsayi olarak islem görür ve 16 tabanli (hexadecimal) sayi olarak döner. (Harfler, küçük harf olur).
X	Degisken tamsayi olarak islem görür ve 16 tabanli (hexadecimal) sayi olarak döner. (Harfler, büyük harf olur).

Her iki fonksiyonun da kullanilis biçimi aynidir:

```
printf( "biçim" , $degisken1, $degisken2, ... "metin" );
```

Burada "biçim" yerine yukaridaki biçim parametlerini yazariz. Biçim parametrelerinin önüne yüzde isareti konur; en fazla bes belirleyici özellik alabilir. Yukaridaki tür belirten biçimlendirme parametlerine ek olarak diger özellikler söyle siralanir:
Doldurma karakteri: tek tirnak ve onu izleyen bir karakterden olusur.
Hizalama: Eksi isaretinin varaligi yazinin sola, yoklugu ise saga hizalanma anlamina gelir.
En az-en çok uzunluk: Sayi-nokta-sayi (örnegin 40.40 gibi) yazilir; birinci sayi azamî, ikinci sayi asgarî uzunlugu belirtir.

Bu üç özellige bir örnek verelim.Bir degiskenin degerinin sonuna yanyana yeteri kadar nokta konarak uzunlugunun 40 karaktere çikartilmasini su deyimle saglariz:

```
$degisken = " Iyilik üzerine " ;
printf( "%'.-40.40s" , $degisken);
```

Burada "%'.-40.40s" seklindeki biçim komutu, Browser penceresinde su görüntüyü olusturur:
"Iyilik üzerine....."
Burada "Iyilik üzerine" degeri 14 karakter oldugu için, sonuna 26 adet nokta eklenmis ve bütün deger sola hizalanmis olacaktir.

Su komut ise iki degiskenin degerini ve verecegimiz bir metni ayni satira yazdiracaktir:

PHP:

```
1.
2.   <?php
3.   $degisken1 = " Iyilik üzerine " ;
4.   $degisken2 = " Iyilik üzerine " ;
5.   $metin = "<br>\n" ;
6.   printf( "%'.-40.40s%'.2d%s" , $degisken1, $degisken2, $metin);
7.   ?>
```

Bu komut Browser penceresinde su görüntüyü olusturur:

ÇIKTI:

```
1.
2.  "Iyilik üzerine.....86"
```

Burada ekledigimiz ikinci "%'.2d" seklineki biçim komutu ile, ikinci degiskenin degeri, en az sifir en çok iki adet nokta ile doldurulmak ve saga hizalanarak ondalik sayi olarak görüntülenmek üzere biçimlendiriliyor. Üçüncü biçim komutu olan "%s" ise üçüncü degiskenin sadece alfanümerik olarak muamele görmesini sagliyor. Biçim komutlarinin arasinda bosluk bulunmamasi, ait olduklari degisken degerlerinin de aralarina bosluk konmamasina sebep oluyor. Üçüncü degiskenin etkisini, kagit üzerinde göremiyoruz; ancak bu Browser penceresinde bundan sonra gelecek unsurlarin bir satir asagi kaymasini saglayacaktir.

Dördüncü biçim özelligi, ondalik sayilarin virgülden (veya nokdtadan) sonra ondalik bölümünün kaç hane olacagini belirler. Bunu da bir örnekle görelim:

PHP:

```
1.
2.   <?php
3.   $degisken = " 124 " ;
4.   printf( "Degeri (ABD) $%.2f" , $degisken);
5.   ?>
```

Bu biçimlendirme komutu da Browser penceresine su yaziyi yazdirir:

ÇIKTI:

```
1.
2.   Degeri (ABD) $124.00
```

str_replace ()

str_replace("<script","<yasak_script",$satir[mesaj]);

number_format()

```
1.
2.   <?php
3.   $degisken = 1234567890.1234567890 ;
4.   echo (number_format($degisken, 4 chr(44) , ".") ); //chr(44)=virgü
5.   ?>
```

Bu deyimle 1234567890.1234567890 seklindeki deger,
Browser penceresine "1.234.567.890,1235" seklinde yazdirilacaktir.

Düzenli Ifadeler

^.+@.+\\..+$

isaretlerinin, Düzenli Ifade islemlerine ait oldugunu belirtilim. Bu isaretler ve onlarin arasina koydugumuz karakter örnekleri ile, PHP'nin aradigimiz bir metnin karakterlerinin hangi dizilis, siralanis konumunda olduguna bakarak, bize o metni bulmasini saglariz; ya bu metni kullaniriz, sileriz veya degistiririz. Dolayisiyla, Düzenli Ifade demek, bir dizilis, siralanis biçimi demektir.

Eslestirme deyimleri ve isaretler
PHP'nin karakter ve siralanis eslemede kullanilan düzenli ifade komutlarini kisaca ele alalim; sonra bunlari kullanmamiza imkan veren fonksiyonlari görelim.

^hakk

"hakk" ile baslayan bütün kelimeleri bulur.

edilemez$

Bu deyim ise PHP'ye "edilemez" ile biten bütün kelimeleri bulur

^hakki$

PHP, basinda ^ isareti, sonunda $ isareti bulunan kararter siralanisini, aynen arar; yani bu deyim, birinci örnekteki üç cümleyi de bulamaz.

Hakk

Bu deyim ise her üç cümleyi de buldurur; çünkü üçünde de bu dört karakter bu siralanisla mevcuttur. PHP'nin Düzenli Ifadeleri, bütün rakam ve harfleri eslestirebilir. Fakat sorun, özel karakterlerde çikar. Sözgelimi, sekme isareti, satir sonlarinda yeni-satir/satirbasi isareti, gibi özel karakterleri, ancak önlerine Escape isareti olan ters bölü isaretini koyarak buluruz.

Düzenli Ifadelerde Özel Karakterler

[\b] Geri (Backspace) karakterini bulur.
\b Belirtilen karakterle sinirlanan kelimeyi bulur: k\b, "hak mücadelesi" ifadesindeki birinci k'yi bulur; çünkü bu harf, bir kelime sinirlayicidir.
\B Belirtilen karakterle sinirlanmayan kelime yoksa, baslayani bulur: k\Bi, "üç kisi" ifadesindeki 'ki'yi bulur.
\cX X yerine yazacagimiz kontrol karakterini bulur. Örnegin, \cA, Ctrl+A'yi, \cZ ise Ctrl+Z'yi bulur.
\d 0'dan 9'ya kadar bir rakami bulur: IE\d, her ikisi de herhangi bir rakamla biten "IE5" ve "IE4" degerlerini ikisini de bulur,
\D Herhangi bir ondalik isaretini bulur.
\f Form-feed (kagit çikart) karakterini bulur.
\n Newline (yeni satir) karakterini bulur.
\r Return (satirbasi) karakterini bulur.
\s Bosluk (space) bulur.
\S Yatay ve düsey sekme, kagit-çikart, yeni satir, satirbasi ve bosluk disindaki herhangi bir karakteri bulur.
\t Yatay sekme (Tab) karakterini bulur.
\v Düsey sekme karakterini bulur.
\w Herhangi bir harf, rakam veya alt-çizgiyi bulur.
\W Harf, rakam ve alt-çizgi disindaki karakteri bulur.
\xHex Verilen 16 tabanli (Hexadecimal) sayiya uygun Escape karakterini bulur. Örnegin, \n25, % isaretini bulur.

Bu arada noktalama isaretlerini arattirirken, önlerine ters bölü isareti koymak gerekir. Ters bömü isaretini de yine önüne ters bölü isareti koyarak (\\) arttirabilirsiniz.

Karakter Gruplari
PHP'nin Düzenli Ifadeleri'nde kolaylik saglayan ve mesela ziyaretçinin bir Form'da bir INPUT etiketine verdigi yanitilarin içinde olmamasi veya olmamasi gereken karakterleri bulmamiza imkan veren karakter gruplari olusturma yöntemini de kullanabiliriz. Sözgelimi bütün sesli hafleri aratmak için söyle bir karakter grubu olusturabiliriz:

[OoUuÖöAaOoEeiIiI]

Karakter gruplarini köseli parantez içinde yazariz. Bu deyimle, PHP, içinde herhangi bir sesli harf bulunan bütün degerleri eslestirecektir. Bu yöntemden yararlanarak, su gruplari kullanabiliriz:

[a-z] Herhangi bir küçük harfi bulur.
[A-Z] Herhangi bir büyük harfi bulur.
[a-zA-Z] Herhangi bir büyük veya büyük harfi bulur.
[0-9] Herhangi bir rakami bulur.
[0-9\.\-] Herhangi bir rakami, noktayi veya kesme çizgisini bulur.
[\f\r\t\n] Herhangi bir Form-feed (kagit çikart), Newline (yeni satir), Return (satirbasi) karakterini veya boslugu (space) bulur.

Sözgelimi, bir alfanümerik deger kümesinde b3, u2, n9 gibi birincisi küçük harf, ikincisi rakam olan iki karakterlik dizileri bulmak istiyorsak, arama grubunu söyle kurariz:

^[a-z][0-9]$

Bu deyim PHP'ye, a'da z'ye küçük harfle baslayan, (^isareti aranan unsurun degerin basinda olmasi gerektigini söylüyor) ve sonunda 0'dan 9'a bir rakam bulunan kelimeleri bulmasini söyleyecektir. PHP, bu kelimenin sadece iki harfli olmasina dikkat edecektir; çünkü grubumuzun bir basi ve bir de sonu belirlendigine göre, üç karakterli degerlerin bulunmasi imkani yoktur.

^ isareti köseli parantez içinde grup deyimi olustururken kullanilirsa, bu olumsuzluk anlami tasir. Sözgelimi, iki rakamli ancak birinci karakteri rakam olmayan fakat ikinci karakteri rakam olan degerlerin bulunmasi için su deyim gerekir:

```
^[^0-9][0-9]$
```

Burada en bastaki ^isareti "basinda" demektir; ancak hemen arkasindan gelen grupta "rakam olmayan" demis oluyoruz; ikinci grup ve sonundaki $ isareti ile "rakamla biten" anlamina geliyor. Deyimde sadece bas ve sonu gösteren iki eslestirme unsuru bulunduguna göre bu deyim, "basinda rakam olmayan, sonunda rakam olan iki karakterli degerleri" bulmaya yarayacaktir. Bu deyim söz gelimi 13'ü bulmayacak, fakat u2'yi bulacaktir. Bu yöntemle su gruplari yapabiliriz:

[^a-z]	Küçük harf olmayan herhangi bir harfi bulur.
[^A-Z]	Büyük harf olmayan herhangi bir harfi bulur.
[^\\/\^]	\ , / veya ^ disinda herhangi bir karakteri bulur.
[^\"\']	Çift ve tek tirnak disinda herhangi bir karakteri bulur.

Grup olusturmada kullandigimiz özel karakterler de vardir. Örnegin nokta isareti (.), yeni satir baslangici olmayan herhangi bir karakter anlamina gelir. Dolayisiyla,

```
^.0$
```

deyimi yeni satirla baslamayan ve sifir ile biten herhangi iki karakterli degeri bulacaktir.

Karakter eslestirmede tekrar sayisi da bir özellik olarak kullanilabilir. Tekrar sayisi belirtmek için süslü parantez ({}) kullaniriz.

Örnekler:

^a{4}$ Içinde sadece dört adet küçük a harfi bulunan kelimeleri seç: aaaa.
^a{2,4}$ Içinde sadece iki üç veya dört adet küçük a harfi bulunan kelimeleri seç: aa, aaa, aaaa gibi
^a{2, } Iki veya daha fazla küçük a harfi bulunan kelimeleri seç: haar, haaar, haaaar gibi. Bu deyim "har" kelimesini seçmez.
\t{2} Ardarda iki sekme isaretini bul
.{2} Herhangi çift karakteri bul: aa, &&, == gibi
^\-{0,1}[0-9]{1,}$ Negatif veya pozitif herhangi bir tam sayiyi bul
^[0-9]{1,}$ Pozitif herhangi bir tam sayiyi bul

Bu tür deyim olusturma islemleri giderek karmasiklasabilir. Örnegin:

^\-{0,1}[0-9]{0, }\.{0,1}[0-9]{0, }$

Bu karmasik deyim aslinda sadece "Negatif veya pozitif bir ondalik (double) degeri bul," anlamina geliyor. Kisaca irdelersek, aranan degerin sifir veya bir kere tekrarlanan bir kesme çizgisiyle baslayabilecegini ("Sifir veya bir kere" demek, olsa da olur, olmasa da anlamina geliyor!) bunu sifir veya daha fazla kere tekrarlanan bir rakamin izleyebilecegini, onu da sifir veya bir kere tekrarlanan bir nokta isareti ile sonunda sifir veya daha fazla kere tekrarlanan herhangi bir rakamin izleyebilecegini söylemis oluyoruz.

PHP bu tür karmasik ifadelerin hatasiz yazilmasini saglayan kisayollara sahiptir. Bunlari siralayalim:

? {0,1} anlamina gelir. Kendisinden önce yer alan unsurun en az sifir en çok bir kere tekrar edilmesi gerektigini (olmayabilecegini ama olursa en fazla bir kere olabilecegini) belirtir.
* {0, } anlamina gelir. Kendisinden önce yer alan unsurun sifir veya daha fazla kere tekrar edilmesi gerektigini (tümüyle opsiyonel oldugunu) belirtir.

+ {1, } anlamina gelir. Kendisinden önce yer alan unsurun en az bir veya daha çok kere tekrar edilmesi gerektigini (bulunmasinin zorunlu oldugunu) belirtir.

Bu kisa-yollari kullanarak, yukaridaki karmasik ifadeleri basitlestirelim:

^[a-zA-Z0-9_]+S En az bir harf veya rakam veya altçizgi içeren herhangi bir kelime
^[0-9]+S Herhangi bir pozitif tamsayi
^\-?[0-9]+S Herhangi bir tamsayi
^\-?[0-9]*\.[0-9*$]+S Herhangi bir kesinli (double) sayi

Bir Düzenli Ifade'nin yazilisinda birden fazla arama-siralanis deyimine yer verebiliriz. Bunu yapmamizi saglayan | isaretidir. Örnegin,

\.com|\.co\.uk

ifadesi ile, ya ".com" ya da ".co.uk" degerlerinin bulunmasini saglayabiliriz. Burada | isareti "veya" kelimesi gibi düsünebilirsiniz.

Düzenli ifadeler yoluyla INPUT etiketinden gelen degerleri incelerken hata yapmak kolaydir. Bunun için kendi ifadelerinizi mutlaka sçesitli olasiliklara karsi sinamalisiniz. Bu bölümün basinda örnek olarak verdigimiz Düzenli Ifade'yi hatirliyor musunuz?

^.+@.+\\..+$

Örnegin bu ifade, ziyaretçinin elektronik posta adresini yazmasi gereken bir INPUT etiketinin sagladigi degerin gerçekten elektronik adres biçimi tasiyip tasimadigini sinar. Bastaki ^ ve nokta isaretleri ile arti isareti degerin önünde bosluk olmamasini sagliyor; @ isareti ise degerin içinde @ bulunmasi gerektigine isaret ediyor. Tekrar eden nokta ve arti isaretleri "ne kadar olursa olsun ve ne olursa olsun" anlamina geliyor. Bunu izleyen nokta karakterini gösteren (\.) isaret buralarda bir de gerçekten nokta olmasi gerektigini ve bunu izleyen nokta ve arti tekrar "ne olursa olsun, ne kadar olursa olsun" anlamini tasiyor. Baska bir deyisle, aradigimiz degerin "herhangi bir sey" @ "herhangi bir sey daha" . "birseyler daha" seklinde oldugunu belirtmis oluyoruz. Ne var ki deyimiçinde iki nokta veya iki @ isareti olan veya @ isareti ile nokta arasinda bir sey bulunmayan veya @ veya noktadan öncesi ya da sonrasi bos olan bütün degerleri safdisi etmeye yetmeyecektir.

Düzenli Ifade Fonksiyonlari

Yukarida ögrendigimiz Düzenli Ifade yazma tekniklerini, PHP'nin bize sagladigi bes fonksiyonda parametre olarak kullaniriz. PHP'nin ayrica Perl-tarzi düzenli ifade fonksiyonlari da vardir. Bu fonksiyonlardan, ya bize bir boolean (dogru/yanlis) deger döner; ya da fonksiyon istedigimiz isi yaparak vardigi sonuçlari verdigimiz degiskene yazar. Biz, daha sonra bu degere bakarak veya degiskenin degerlerini kullanarak, PHP programimizin akisini kontrol edebiliriz. Burada ele alacagimiz fonksiyonlara iliskin örneklerde, daha önceki bölümlerde olusturdugumuz konuk defteri programi ile Web ziyaretçilerimizin sunucuya gönderecegi bilgileri dogrulamaya ve muhtemel zararli kodlardan ayiklamaya çalisacagiz.

ereg() ve eregi()

PHP'nin temel Düzenli Ifade Fonksiyonu, ereg(), arattigimiz karakter siralanisi bulundugu taktirde dogru, bulamadigi taktirde yanlis karsiligi bir deger verir. Fonksiyonu söyle yazariz:

```
$bir_degisken = ereg("eslestirilecek_sira" , $kaynak , $yeni_degisken);
```

Fonksiyonun aradigimiz eslestirmeyi yapmasi halinde, buradaki $bir_degisken'in degeri true/dogru, yapamamasi halinde false/yanlis olacaktir. Eslestirme sirasinin nasil olusturuldugunu yukarida gördük; bu ifadelerden isimize uygun olani buraya tirnak içinde yazariz. $kaynak, eslestirilecek siralamanin içinde aranacagi degeri tutan degiskendir. Fonksiyonun bir diger becerisi, eger eslestirilecek siralamayi gruplar halinde verirsek, kaynakta yapacagi eslestirme olursa, buna uygun degerleri bir dizi degiskene yazabilmesidir; istersek bir parametre olarak bu yeni degiskenin almasini istedigimiz adi veririz; böylece eslestirme sonucu bulunan degerler kaydedilmis olur.

eregi(), aynen ereg() fonksiyonu gibi çalisir; sadece eslestirecegi degerlerde büyük-harf/küçük-harf farki gözetmez.

Daha önceki bölümde olusturdugumuz ve kd_01.php adiyla kaydettigimiz konuk defteri programinin akis planini, ziyaretçinin Form'a yazdigi ve sunucuda $HTTP_POST_VARS dizi-degiskeninde tutulan degiskenlerinden elektronik posta adresi ilge ilgili olani gerçekten içinde en az bir @ isareti ile en az bir adet nokta içip içermedigine bakarak sinayabiliriz. Böyle bir sinama için gerekli kod söyle olabilir:

```
1.
2.    <?php
3.    if (eregi("^.+@.+\\..+$", $adres, $email)) {
4.       }
5.       else {
6.       $hata = "Elektronik posta adresinizde bir hata var!<br>";
7.             echo $hata;
8.             include("kd_hata_halinde.htm");
9.             exit;
10.      }
11    ?>
```

Program, bu örnekte $adres degiskeninde kayitli degerin içinde aradigi siralamayi bulursa, eslesen degeri $email adli yeni bir degiskene yazacak ve if sinavinin sonucu dogru olacaktir. Bu siralamaya uygun bir deger bulunamazsa, if sinavi else deyimine atlayacak ve bir hata mesaji üretilerek, bu program durdurulacaktir.

ereg_replace() ve eregi_replace()

Gördügümüz gibi, ereg() arattigimiz karakter siralanisi bulundugu taktirde dogru, bulamadigi taktirde yanlis karsiligi verdikten sonraduruyor! Oysa kimi zaman arattigimiz ve bulunan degerin baska bir degierle degistirilmesi gerekebilir. Bunun için ereg_replace() ve eregi_replace() fonksiyonlarini kullaniriiz:

ereg_replace("eslestirilecek_sira" , yeni_metin , $kaynak);

Fonksiyonun aradigimiz eslestirmeyi bulursa, bu degerin yerine verdigimiz yeni metni koyacaktir; yeni metni bir degiskenin degeri olarak da verebiliriz. Uygulama örnegi için yine konuk defteri örnegine dönelim. Ziyaretçilerimiz kimi zaman yanlislikla, kimi zaman pek de iyi niyet sonucu olmadan, kendilerinden bekledigimiz isim, adres ve mesaj yerine sunucu veya baska ziyaretçilerin Browser programlari tarafindan kod gibi algilanacak metinler yazabilirler. Burada sadece bu tür zararli metinlerin genellikle programlarda bulunmasi gereken karakterler içerdigini söylemekle yetinelim. Bu tür karakterlerin basinda < ve > isaretleri bulunur! Dolayisiyla, biz de ziyaretçimizden gelecek verilerin yazildigi degiskenlerin degerlerinde bu isaretleri aratabilir ve bunlari içi bos bir alfanümerik deger ile degistirebilir; yani silebilir. Zararli olabilecek kodlarin arasinda daha bir çok karakter bulunabilir; ancak Script diliyle yazilmasi gereken bu kodlardan < ve > isaretlerini kaldirilmasi kodlari islemez hale getirecegi için, su asagidaki örnek yeterli olabilir:

```
1.
2.   <?php
3.      $adi = ereg_replace("<","",$adi);
4.      $adi = ereg_replace(">","",$adi);
5.      $adres = ereg_replace("<","",$adres);
6.      $adres = ereg_replace(">","",$adres);
7.      $mesaj = ereg_replace("<","",$mesaj);
8.      $mesaj = ereg_replace(">","",$mesaj);
9.   ?>
```

Burada ereg_replace() fonksiyonu, ziyaretçiden gelecek üç degiskenin degerlerinde < ve > isaretlerini aramakta onlarin yerine içi bos bir metin ("") yazmaktardir.

split()

Düzenli Ifade ile çalisan bu fonksiyon, verecegimiz eslestirme siralamasini sinirlayici olarak kullanarak, belirtecegimiz degerde buldugu deger parçalarini ayirir ve bunlari ayri ayri bir dizi degiskenin elemanlari olarak kaydeder. Bu fonksiyonu söyle yazariz:

$yeni_dizi_degisken = split("eslestirilecek_sira" , $kaynak, sinir_sayisi);

Fonksiyon, aradigi siralamayi bulamazsa, false/yanlis sonucunu verir. Burada sinir sayisi olarak verecegimiz rakam, olusturulacak yeni dizi degiskene en fazla kaç eleman yazilmasini istedigimizi gösterir. Bu sayiyi vermezsek, PHP yeni dizi degiskenin gerektigi kadar elemana sahip olmasini saglar. Bir örnek vererek, bu fonksiyonu nasil kullanibilecegimizi görelim:

```
1.
2.  <?php
3.  $metin = "Insan sözüyle kendini gösterir, davranislariyla ruh halini aksettirir.";
4.  $aranan = " ";
5.  $yeni_dizi_degisken = split($aranan, $metin);
6.  foreach ($yeni_dizi_degisken as $eleman) {
7.  print "$eleman <br>";
8.  }
```

Bu programda PHP, $metin degiskeninin içerdigi degerde $aranan degiskeninin içerdigi degeri, yani boslugu, eslestirilecek unsur olarak kullanacak ve $metin degiskeninin degerini bosluklarindan parçalara ayiracaktir. Ayrilicak her yeni parça, $yeni_dizi_degisken adli degiskenin elemanlari olarak atanacaktir. Programin geri kalan kismi ise, bu yeni dizinin elemanlarini görüntülemekterdir.

sql_regcase()

Içinde büyük harf-küçük harf ayrimi olan bir degeri büyük harf-küçük harf ayrimi olmayan Düzenli Ifadeler haline çevirir. Bu fonksiyon bizden Düzenli Ifade almaz, tersine Düzenli Ifade olusturur. Örnek:

```
1.
2.  <?php
3.  $metin = "Sözler";
4.  echo(sql_regcase($metin);
5.  ?>
```

Bu program, Browser penceresine su metni yazdirir:

ÇIKTI:

```
1.
2.  [Ss][Öö][Zz][Ll][Ee][Rr]
```

Tarih ve saat Verisi
PHP, o andaki zaman bilgisini, saat, dakika, saniye ve salise olarak; tarih bilgisini yil, ay, gün (sayi veya isim olarak), programimizin herhangi bir yerinde bize bildirebilir. Bu bilgiyi Web sunucusunda istedigimiz anda, muhtemelen sunucunun bulundugu bilgisayarin sistem saatinden alacak olan PHP, sunucu programinda farkli bölgesel ayarlar için gerekli düzenleme yapilmissa, bu imkandan yararlanarak bize sunucunun degil, arzu ettigimiz bölgenin saat ve tarihini bildirebilir.

Özellikle Türkiye'de olmayan bir sunucuda bu imkanin bulunup bulunmadigini, ancak sinayarak veya sistem yöneticisine sorarak ögrenebiliriz. Böylebir siNama için su kodlari programinizin basina koyun:

```
1.
2.   <?php
3.   setlocale ("LC_TIME", "TR");
4.   print (strftime ("Türkçe bugün günlerden: %A "));
5.   ?>
```

Browser penceresinde "Türkçe bugün günlerden Sunday" yazisini okursaniz, sunucuda Türkçe için bölgesel ayar destegi yok demektir!

PHP'nin zaman ve tarih belirlemekte kullanabileceginiz baslica fonksiyonu getdate() ise söyle kullanilir.

getdate()

Tarih ve saat bilgisini alir ve ve vereceginiz bir isimdeki dizi-degiskende kaydeder. Örnek:

```
$saat_tarih = getdate()
```

Bu durumda, $saat_tarih dizi degiskeninde sirasiyla su bilgiler yer alir:

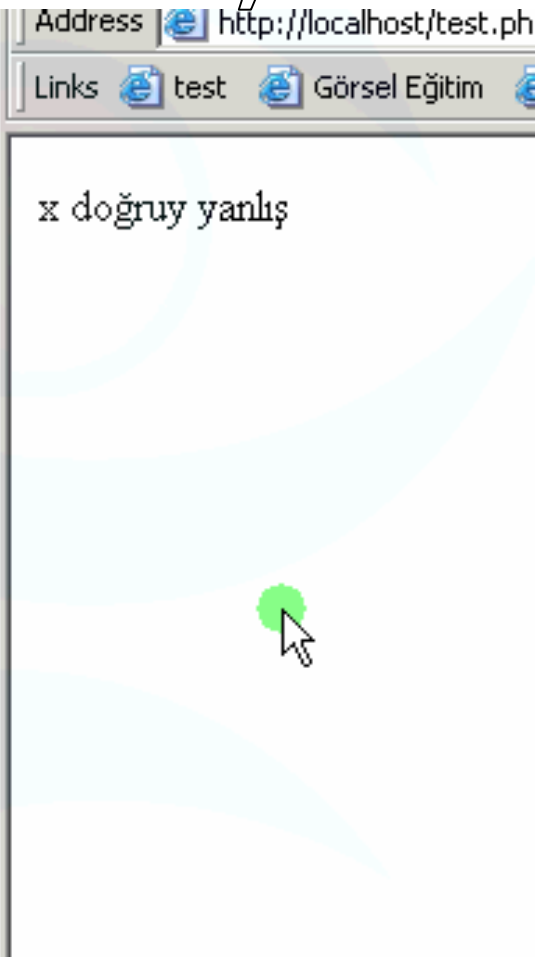
32	saniye
57	dakika
6	saat
30	ayin gün sayisi (1-31)
0	haftanin gün sayisi (1-7)
7	ayin sayisi (1-12)
2000	yil
211	yilin kaçinci günü
Sunday	günün adi
July	ayin adi
964929452	Unix sistemlerinde Epoch biçiminde zaman bilgisi

Escaped characters

Özel karakter	Anlamı
\n	linefeed (LF or 0x0A (10) in ASCII) - Yeni Satır (New Line)
\r	carriage return (CR or 0x0D (13) in ASCII) - Satır Başı
\t	horizontal tab (HT or 0x09 (9) in ASCII) - Sekme (Tab) karakteri
\\	backslash - Ters Bölü işareti
\$	dollar sign - $ işareti
\" , \'	double-quote - Çift tırnak işareti , Single-quote - Tek Tırnak işareti
\[0-7]{1,3}	the sequence of characters matching the regular expression is a character in octal notation
\x[0-9A-Fa-f]{1,2}	the sequence of characters matching the regular expression is a character in hexadecimal notation

Veri Türleri (Boolean)

```
2
3  $x = "TRUE";
4  $y = 0 ;
5
6
7  if ($x=true) {
8  echo " x doğru";
9
10 }
11 if ($y) {
12 echo "evet y = 2,doğru";
13    }
14    else {
15    echo "y yanlış";
16    }
17
18
19
20 ?>
```



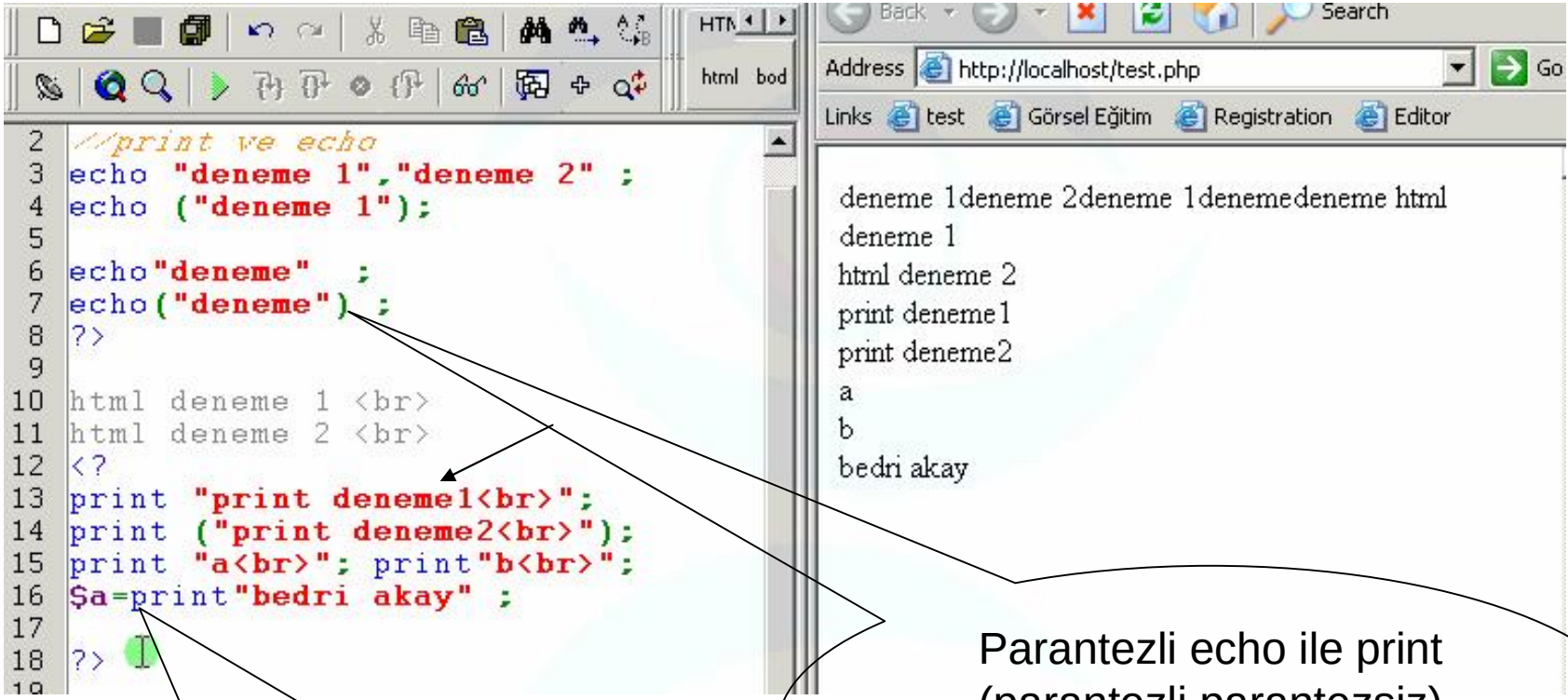
Veri Türleri

```php
<?php

$a = 1234567890;
$a =21474836478;
$a = 0123; //octal 0-7
$a = 0x12FFFF; // hexadecimal 0-9
$a= -2569879 ;
print gettype($a)    ;

?>
```

```php
<?php

$a = -12.50; // 1.8e308 , 14
$a = 15.00 + 45.00;
$a = -1.25e6; // 1.25 on üzeri 6
$a = 5e-3   ;
print number_format($a,10)    ;

?>
```

Links test Görsel Eğ

0.0050000000

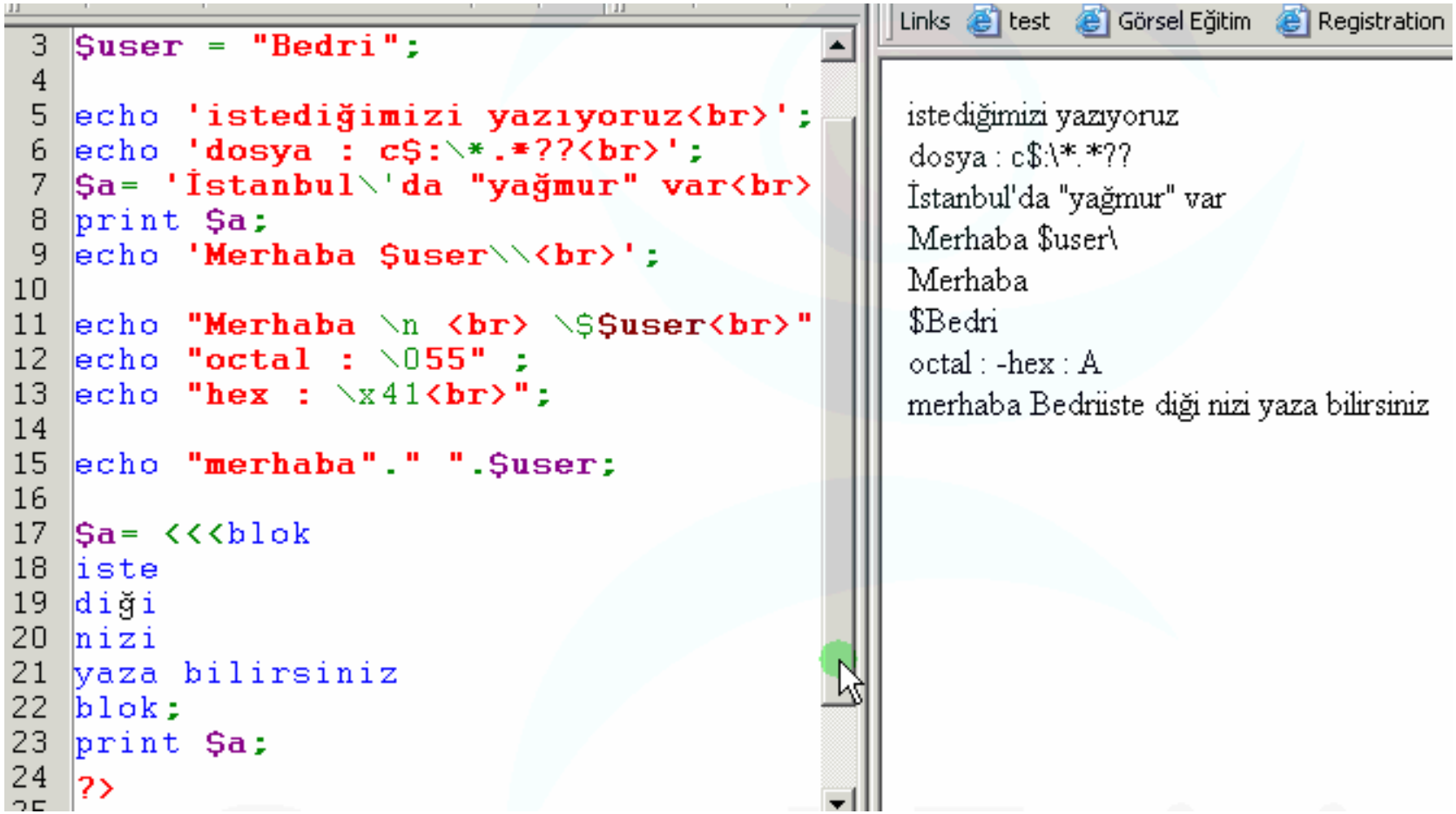


```
2  //print ve echo
3  echo "deneme 1","deneme 2" ;
4  echo ("deneme 1");
5
6  echo"deneme"  ;
7  echo("deneme") ;
8  ?>
9
10 html deneme 1 <br>
11 html deneme 2 <br>
12 <?
13 print "print deneme1<br>";
14 print ("print deneme2<br>");
15 print "a<br>"; print"b<br>";
16 $a=print"bedri akay" ;
17
18 ?>
```

```
deneme 1deneme 2deneme 1denemedeneme html
deneme 1
html deneme 2
print deneme1
print deneme2
a
b
bedri akay
```

Parantezli echo ile print (parantezli parantezsiz) komutu birden fazla (virgülle ayrılmış) parametre kabul etmezler

echo ya nazaran print komutu değer döndürür (yazdırma başarılıysa doğru, else false)



```
3  $user = "Bedri";
4
5  echo 'istediğimizi yazıyoruz<br>';
6  echo 'dosya : c$:\*.*??<br>';
7  $a= 'İstanbul\'da "yağmur" var<br>
8  print $a;
9  echo 'Merhaba $user\\<br>';
10
11 echo "Merhaba \n <br> \$$user<br>"
12 echo "octal : \055" ;
13 echo "hex : \x41<br>";
14
15 echo "merhaba"." ".$user;
16
17 $a= <<<blok
18 iste
19 diği
20 nizi
21 yaza bilirsiniz
22 blok;
23 print $a;
24 ?>
```

Links test Görsel Eğitim Registration

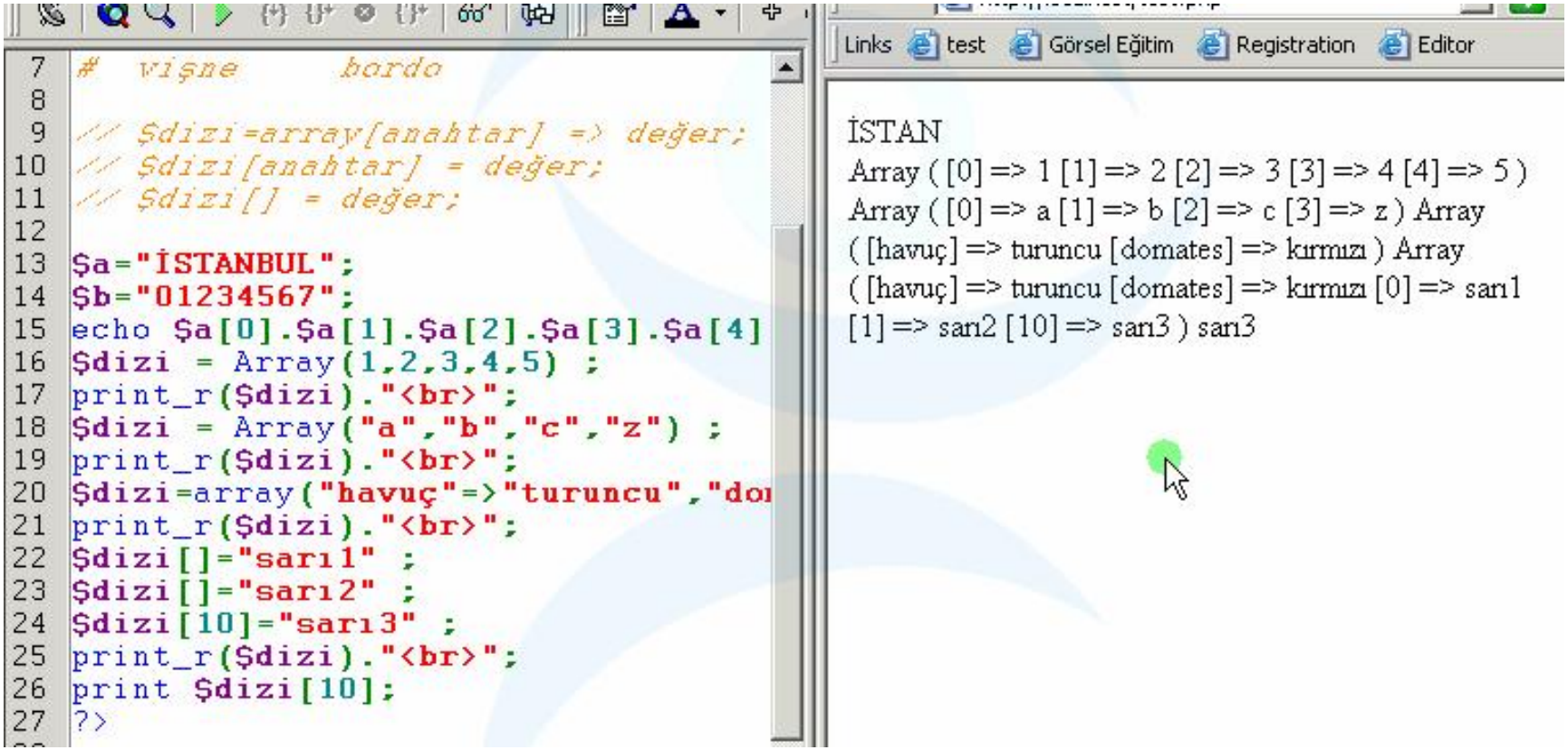
istediğimizi yazıyoruz
dosya : c$:*.*??
İstanbul'da "yağmur" var
Merhaba $user\
Merhaba
$Bedri
octal : -hex : A
merhaba Bedriiste diği nizi yaza bilirsiniz

diziler

```php
<?php

#  ANAHTAR    DEĞER
#  havuç      turuncu
#  domates    kırmızı
#  biber      yeşil
#  vişne      bordo

// $dizi=array(anahtar) => değer;
// $dizi[anahtar] = değer;
// $dizi[] = değer;

$a="İSTANBUL";
$b="01234567";
echo $a[0]."<br>";
echo $a[1]."<br>";
echo $a[2]."<br>";
echo $a[3]."<br>";
echo $a[4]."<br>";


?>
```

Links test

İ
S
T
A
N



```
7  #  vişne      bordo
8
9  // $dizi=array[anahtar] => değer;
10 // $dizi[anahtar] = değer;
11 // $dizi[] = değer;
12
13 $a="İSTANBUL";
14 $b="01234567";
15 echo $a[0].$a[1].$a[2].$a[3].$a[4]
16 $dizi = Array(1,2,3,4,5) ;
17 print_r($dizi)."<br>";
18 $dizi = Array("a","b","c","z") ;
19 print_r($dizi)."<br>";
20 $dizi=array("havuç"=>"turuncu","do
21 print_r($dizi)."<br>";
22 $dizi[]="sarı1" ;
23 $dizi[]="sarı2" ;
24 $dizi[10]="sarı3" ;
25 print_r($dizi)."<br>";
26 print $dizi[10];
27 ?>
```

Links test Görsel Eğitim Registration Editor

İSTAN
Array ([0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5)
Array ([0] => a [1] => b [2] => c [3] => z) Array
([havuç] => turuncu [domates] => kırmızı) Array
([havuç] => turuncu [domates] => kırmızı [0] => sarı1
[1] => sarı2 [10] => sarı3) sarı3

Devam edecek...