

Php Nedir?

Web Programlama Dersi

PHP nedir?

- PHP, sunucu tarafı bir betik dilidir. Statik web siteleri veya Dinamik web siteleri veya Web uygulamaları geliştirmek için kullanılır. PHP, daha önce Kişisel Ana Sayfalar anlamına gelen Köprü Metni Ön İşlemcisi anlamında kullanıldı.
- PHP betikleri(kodları) yalnızca PHP'nin kurulu olduğu bir sunucuda yorumlanabilir.
- PHP betiklerine erişen istemci bilgisayarlar yalnızca bir web tarayıcısı gerektirir.
- Bir PHP dosyası PHP etiketleri içerir ve “.php” uzantısıyla biter.

Komut Dosyası Dili nedir?

- Komut dosyası, çalışma zamanında yorumlanan bir dizi programlama talimatıdır.
- Komut dosyası dili, komut dosyalarını çalışma zamanında yorumlayan bir dildir. Komut dosyaları genellikle diğer yazılım ortamlarına gömülür.
- Komut dosyalarının amacı, genellikle bir uygulamanın performansını artırmak veya rutin görevleri gerçekleştirmektir.
- Sunucu tarafı komut dosyaları sunucuda yorumlanırken, istemci tarafı komut dosyaları istemci uygulaması tarafından yorumlanır.
- PHP, sunucuda yorumlanan bir sunucu tarafı komut dosyası iken [JavaScript](#) , istemci tarayıcısı tarafından yorumlanan bir istemci tarafı komut dosyası örneğidir. Hem PHP hem de JavaScript, HTML sayfalarına gömülebilir.

PHP'nin açılımı nedir?

- PHP - **Kişisel Ana Sayfa** anlamına gelir , ancak şimdi ; PHP: Köprü Metni Ön İşlemcisi anlamında kullanılıyor.
- PHP kodu, HTML koduna gömülebilir veya çeşitli web şablon sistemleri, web içerik yönetim sistemi ve web çerçeveleri ile birlikte kullanılabilir.
- PHP Sözdizimi;
 - **<?php**
 - **Echo "Hello World";**
 - **?>**
- Bir PHP dosyası, HTML gibi etiketler ve JavaScript gibi istemci tarafı komut dosyaları da içerebilir.
- PHP Dili öğrenirken **HTML ek bir avantajdır** . PHP'yi HTML bilmeden de öğrenebilirsiniz, ancak en azından HTML'nin temellerini bilmeniz önerilir.
- Veritabanı destekli uygulamalar için veritabanı **yönetim sistemleri DBMS**.
- Etkileşimli uygulamalar ve web hizmetleri gibi daha gelişmiş konular için **JavaScript'e** ihtiyacınız olacak .

PHP'yi neden kullanmalı?

- PHP **açık kaynak kodlu ve ücretsizdir.**
- JSP, ASP vb. gibi diğer dillere kıyasla kısa öğrenme eğrisi.
- Büyük topluluk belgesi
- Çoğu web barındırma sunucusu, ASP gibi IIS'ye ihtiyaç duyan diğer dillerin aksine PHP'yi varsayılan olarak destekler. Bu, PHP'yi uygun maliyetli bir seçim haline getirir.
- PHP ile elde edeceğiniz diğer bir avantaj, onun bir **sunucu tarafı betik dili** olmasıdır ; bu, yalnızca sunucuya yüklemeniz gerektiği anlamına gelir ve sunucudan kaynak isteyen istemci bilgisayarlarda PHP'nin kurulu olması gerekmez; sadece bir web tarayıcısı yeterli olacaktır.
- PHP, **MySQL ile el ele çalışmak için yerleşik desteğe** sahiptir ; bu, PHP'yi diğer veritabanı yönetim sistemleriyle kullanamayacağınız anlamına gelmez. PHP ile aşağıdaki veritabanlarını da kullanabilirsiniz;
 - Postgres
 - oracle
 - MS [SQL](#)
- PHP **çapraz platformdur**; bu, uygulamanızı Windows, Linux, Mac OS vb. gibi bir dizi farklı işletim sistemine dağıtabileceğiniz anlamına gelir.

PHP Dosya Uzantıları

- Dosya Uzantısı ve Etiketleri Sunucunun PHP dosyalarımızı ve betiklerimizi tanımlayabilmesi için dosyayı “ .php ” uzantılı olarak kaydetmeliyiz .
- PHP, HTML ile çalışmak üzere tasarlanmıştır ve bu nedenle HTML koduna gömülebilir.
- Herhangi bir html etiketi olmadan PHP dosyaları oluşturabilirsiniz ve buna Pure PHP dosyası denir.
- Sunucu PHP kodunu yorumlar ve sonuçları web tarayıcılarına HTML kodu olarak gönderir.
- Sunucunun PHP kodunu HTML kodundan ayırabilmesi için, PHP kodunu her zaman PHP etiketleri içine almalıyız. **<?php ?>**
- Bir PHP etiketi, "küçüktür" sembolüyle başlar, ardından soru işareti ve ardından “php” kelimeleri gelir. **<?php**
- PHP büyük/küçük harf duyarlı bir dildir, “**VAR**”, “**var**” ile aynı değildir.
- PHP etiketlerinin kendileri büyük/küçük harfe duyarlı değildir, ancak küçük harf kullanmamız şiddetle tavsiye edilir. Aşağıdaki kod yukarıdaki noktayı göstermektedir.
- ECHO ile **echo** aynıdır.
- **IF** ile **if** aynıdır.
- **ARRAY** ile **array** aynıdır. vs.

PHP Merhaba dünya

- Aşağıda gösterilen program, “Merhaba Dünya!” sözcüklerini veren temel bir PHP uygulamasıdır.
- `<?php`
 - `echo "Merhaba dünya";`
- `?>`
- Yukarıdaki php dosyası bir web tarayıcısında görüntülenirse aşağıdaki çıktıyı üretir.
- Merhaba dünya
- **Özet**
- PHP, Köprü Metni ön işlemcisi anlamına gelir
- PHP, sunucu tarafı bir betik dilidir. Bu, sunucuda yürütüldüğü anlamına gelir. İstemci uygulamalarının PHP'nin kurulu olması gerekmez.
- PHP dosyaları “.php” dosya uzantısıyla kaydedilir ve PHP geliştirme kodu etiketler içine alınır.
- PHP açık kaynak ve çapraz platformdur. Yani tüm işletim sistemlerinde çalışır.

XAMPP nedir?

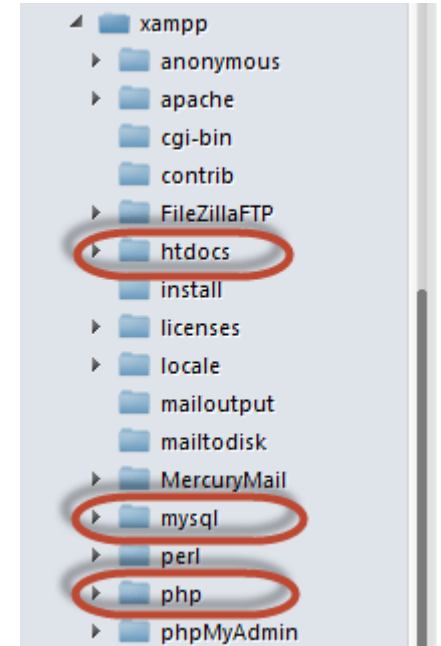
- **XAMPP , bir web sunucusu, MySQL veritabanı motoru ve PHP ve [Perl](#)** programlama paketlerinden oluşan açık kaynaklı, platformlar arası bir web sunucusudur . Apache tarafından derlenir ve korunur.
- Kullanıcıların bilgisayarlarında yerel bir web sunucusu kullanarak php ile dinamik web siteleri oluşturmalarına olanak tanır. Windows, Linux ve Mac'i destekler.
- Apache tarafından derlenir ve korunur. XAMPP kısaltması;
- [X – \[çapraz platform işletim sistemleri\] herhangi bir OS Mac OS , Windows , Linux vb. üzerinde çalışabileceği anlamına gelir .](#)
- A – [Apache](#) – bu web sunucusu yazılımıdır.
- M – MySQL – Veritabanı.
- P – [PHP](#)
- P – Perl – betik dili

XAMPP neden kullanılır?

- XAMPP, Apache, MySQL ve diğer programları komut kullanmadan yönetmek için kullanımı kolay bir kontrol paneli sağlar. PHP'yi kullanmak için Apache ve MySQL'i kurmamız gerekiyor. Diğer şeylerin yanı sıra PHP ve Perl ile kurulması ve entegre edilmesi gerektiği için Apache'yi kurmak ve yapılandırmak kolay değildir. XAMPP, Apache'yi PHP ve Perl ile kurmanın ve entegre etmenin tüm karmaşıklığıyla ilgilenir.
- PHP'nin çalışması için yalnızca bir web sunucusu gerekir. Ancak veri tabanı kullanmak istenirse mysql gibi bir veritabanı kurmak da gerekir.

XAMPP Web Sunucusundaki bazı klasörler

- Bir XAMPP kurulumunda, Windows'ta C sürücüsüne XAMPP yüklediğinizi varsayarsak aşağıda bilmeniz gereken temel dizinlerin bir listesi bulunmaktadır.
- **htdocs** ; bu web kök dizinidir. Tüm PHP projelerimiz/kodlarımız bu dizine yerleştirilecektir.
- **mysql** – bu dizin MySQL veritabanı motoruyla ilgili tüm bilgileri içerir, varsayılan olarak 3306 numaralı bağlantı noktasında çalışır.
- **php** – bu dizin PHP kurulum dosyalarını içerir. **php.ini** adında önemli bir dosya içerir. Bu dizin, PHP'nin sunucunuzda nasıl davranacağını yapılandırmak için kullanılır.
- Varsayılan olarak, Apache web sunucusu 80 numaralı bağlantı noktasında çalışır .



PHP'yi HTML içine gömmek

- PHP dosyaları, uzantılı düz metin dosyalarıdır .php. Bir PHP dosyasının içine, normal HTML sayfalarında yaptığınız gibi HTML yazabilir ve sunucu tarafında yürütme için PHP kodlarını gömebilirsiniz.
- `<!DOCTYPE html>`
- `<html lang="en">`
- `<head>`
- `<meta charset="UTF-8">`
- `<title>A Simple PHP File</title>`
- `</head>`
- `<body>`
- `<h1>`
- `<?php echo "Hello, world!"; ?>`
- `</h1>`
- `</body>`
- `</html>`

Yandaki örnek, iyi biçimlendirilmiş dinamik web sayfaları oluşturmak için PHP kodlarını HTML içine nasıl gömebileceğinizi gösterir. Ortaya çıkan web sayfasının kaynak kodunu tarayıcınızda görüntülerseniz, göreceğiniz tek fark PHP kodunun `<?php echo "Hello, world!"; ?>` "Merhaba, dünya!" çıktısı ile değiştirilmiştir.

Burada ne oldu? Bu kodu çalıştırdığınızda, PHP motoru etiketler arasındaki talimatları yürütür `<?php ... ?>` ve geri kalan şeyi olduğu gibi bırakır. Sonunda web sunucusu, nihai çıktıyı tamamen HTML olan tarayıcınıza geri gönderir.

PHP Echo kullanımı

- Çıktıyı bir web tarayıcısında görüntülemek için kullanılır.
- echo deyimi bir veya daha fazla karakter dizisi yazdırabilir.
- Genel anlamda, echo deyimi, tarayıcıda görüntülenebilen dize(metin), sayılar, değişken değerleri, ifadelerin sonuçları vb. gibi her şeyi görüntüleyebilir.
- Echo'yu parantez olmadan kullanabilirsiniz, örneğin echo veya echo() gibi. Ancak, echo ile birden fazla parametre yazdırmak istiyorsanız, parametreler parantez içine alınmamalıdır.

Echo ile ilgili örnekler

```
<?php
// metin yazdırma
echo "Hello World!";
?>
```

```
<?php
// html kodu yazdırma
echo "<h4>Bu bir başlıktır.</h4>";
echo "<h4 style='color: red;'>
Bu stilli bir başlıktır.</h4>"; ?>
```

```
// değişken tanımlama
$txt = "Hello World!";
$num = 123456789;
$colors = array("Red", "Green",
"Blue");

// Değişkenleri görüntüleme
echo $txt;
echo "<br>";
echo $num;
echo "<br>";
echo $colors[0];
?>
```

Echo kullanımı ile ilgili örnekler

- `<?php`
 - `$s1=43;`
 - `$s2=34;`
 - `$sonuc=$s1+$s2;`
 - `echo "
";`
 - `echo "
 $sonuc"; // 77 yazar`
 - `echo "
";`
 - `echo "Sonuc=", $sonuc; // Sonuc=77 yazar`
 - `echo "
";`
 - `echo "Sonuc=$sonuc"; // Sonuc=77 yazar`
 - `echo "
";`
 - `echo "Sonuc=", $s1+$s2; // Sonuc=77 yazar`
 - `echo "
";`
 - `echo "Sonuc=$s1+$s2"; // Sonuc=43+34 yazar. Çünkü aritmetik işlemler çift tırnak içinde yapılamaz`
 - `echo "
";`
 - `echo "$s1+$s2=", $s1+$s2; // 43+34=77 yazar`
- `?>`

Echo ile yazdırma işlemi Tek Tırnak / Çift Tırnak kullanımı.

Bu da PHP de “ (çift tırnak) ile ‘ (tek tırnak) arasında bazı kullanım farklılıkları olması. Tek tırnak ile vereceğimiz değerlerin içerisine değişken yazamayız. Tek tırnak içine ne yazarsak aynısı ekrana yazılır. Değişkenleri yazdıracaksak çift tırnak kullanmak gerekir.

Çift tırnak kullanılırsa

- <?php
- \$gun=15;
- \$ay="Mart";
- \$yil=2016;
- echo "\$gun/\$ay/\$yil";
- ?>
- Ekrana; Bugün 15/Mart/2016 yazar

Tek tırnak kullanılırsa

- <?php
- \$gun=15;
- \$ay="Mart";
- \$yil=2016;
- echo 'Bugün \$gun/\$ay/\$yil';
- ?>
- Ekrana; Bugün \$gun/\$ay/\$yil yazar

PHP'de Büyük/Küçük Harf Duyarlılığı

- PHP'deki değişken adları büyük/küçük harfe duyarlıdır. Sonuç olarak değişkenler \$color, \$Color ve \$COLOR üç farklı değişken olarak kabul edilir.
- Ancak anahtar sözcükler, işlev ve sınıf adları büyük/küçük harfe duyarsızdır. Sonuç olarak gettype() veya GETTYPE() aynı sonucu verir ve sonuç olarak ekranda 2 tane string ifadesi çıkar.

```
<?php
$color = "blue";
echo gettype($color) . "<br>";
ECHO GETTYPE($color) . "<br>";
?>
```

```
1. <?php
2. $color = "blue";
3. echo gettype($COLOR) . "<br>";
4. ECHO GETTYPE($Color) . "<br>";
5. ?>
```

3. Ve 4. satırlarda hata oluşur. Çünkü 3 farklı \$color değişkeni var. İkisi tanımlanmamış.

Notice: Undefined variable: COLOR şeklinde hata oluşur

İfadeler noktalı virgülle sonlanır

- PHP'de bir ifade , arkasından noktalı virgül (;) gelen herhangi bir ifadedir. PHP etiketleri arasında yer alan herhangi bir geçerli PHP ifadesi dizisi, geçerli bir PHP programıdır. PHP'deki tipik bir ifade, bu durumda \$greeting adlı bir değişkene bir karakter dizisi atamaktadır -
- \$greeting = "Welcome to PHP!";

Parantezler blok yapar

- İfadeler, ifadeler gibi birleştirilemese de, her zaman bir ifadenin gidebileceği herhangi bir yere, bir dizi küme parantezi içine alarak bir ifade dizisini koyabilirsiniz.
- Burada her iki ifade de eşdeğerdir -
- if (3 == 2 + 1)
- { print("Good - I haven't totally");
- print("lost my mind.
");
- }

PHP Değişkenler

- Değişken, çalışma zamanında verileri depolayan bir bellek konumuna verilen addır.
- Değişkenler programın akışı içinde değer saklamak için kullanılırlar.
- Bu değerler metinler, sayılar yada diziler olabilir ve program akışı içerisinde istediğiniz kadar kullanıp değerini değiştirebilirsiniz.
- **PHP'de kullanılan değişkenlerin önünde \$ işareti kullanılır.**
- Bir değişkeni su içeren bir bardak olarak düşünün. Bardağa su ekleyebilir, hepsini içebilir, tekrar doldurabilirsiniz vb.
- Aynısı değişkenler için de geçerlidir. Değişkenler, verileri depolamak ve gerektiğinde depolanan verileri sağlamak için kullanılır. Diğer programlama dillerinde olduğu gibi PHP de değişkenleri destekler.

Sabit ve deęişken tanımlaması Örneęi

- <?php
- \$oyuncuSayisi=11; //sabit
- \$hakemSayisi=3; //sabit
-
- \$skor = NULL; //deęişken
- \$cezaSayisi= NULL; //deęişken
- ?>



FUTBOL

Sayı	Tür	Açıklama
11	Sabit	Her takımdaki oyuncu sayısı
4	Sabit	Maçtaki hakem sayısı
90	Sabit	Maçın bitişı için dakika sayısı
45	Sabit	Her bir devre bitişı için dakika sayısı
3	Sabit	Yapılabilecek oyuncu deęişikliği sayısı

Sayı	Tür	Açıklama
Skor	Deęişken	Maçta atılması muhtemel gol sayısı
Ceza	Deęişken	Maçta gösterilecek kart sayısı
Zaman	Deęişken	Maç içerisinde geçen zaman deęeri
Pas	Deęişken	Maçta atılacak pas sayısı

Değişken tanımlama kuralları

- Bir değişkenin adı harfle yada _ ile başlamalıdır.
- Değişken isimlerinde boşluk bırakılmamalıdır. Boşluğun yerine _ işareti kullanabilirsiniz.
- Bir değişkenin isminde a-z, A-Z, 0-9 ve _ karakterlerinin dışında başka bir karakter kullanamazsınız.
- **Not:** PHP dilinde büyük küçük harf ayrımı olduğundan **\$adi** ile **\$Adi** değişkenlerinin aynı *olmadığını* unutmayınız.
 - <?php
 - \$text="Merhaba Dünya!";
 - \$Text="Merhaba Kütahya!";
 - \$adi_soyadi="Ahmet Can";
 - \$sayi_1=43;
 - \$sayi_2=34;
 - echo \$Text; // sayfaya **Merhaba Kütahya!** Yazar.
 - ?>

Sabitler

- Değeri değiştirilemeyen değişkenlerdir. Bunlar önceden tanımlanır ve program akışında sadece kullanılırlar. Değerleri değiştirilemez. Bir sabit **define(değişkenadı, değeri);** şeklinde tanımlanır.
- <?php
- Define("snc","İşlem sonucu :"); //string bir sabit tanımladık. Adı; snc
- define("pi",3.14); // sabitler kullanılırken önüne \$ işareti konulmaz ve adı çift tırnak içinde tanımlanır. Pi adında sayısal başka bir sabit tanımladık.
- \$alan = pi * 5 * 2; //yarıçapı 5cm olan çemberin alanını hesaplar.
Alan=Pi*r²
- echo pi; // ekrana 3.14 yazar.
- echo snc.\$alan; //ekranda; İşlem sonucu :31.4 yazar
- ?>

sabit değişkenler tanımlanırken ve kullanılırken önüne \$ işareti konulmaz

- `<?php`
- `define("pi",3.14); //`
- `echo $pi; //`
- `pi=3; //`
- `$alan=pi*2*2; //`
- `?>`

Sabit Tanımlama Örneği

- `<?php`
- `// SITE_URL` adında bir Sabit tanımlı.
- `define("SITE_URL", "https://www.tutorialrepublic.com/");`
-
- `// program içinde Sabitin kullanılması`
- `echo 'Ziyaret ettiğiniz için TEŞEKKÜRLER ' . SITE_URL;`
- `?>`
- **İpucu:** Değeri bir değişken yerine bir sabitte saklayarak(örnekte olduğu gibi), uygulamanız çalıştığında değerin yanlışlıkla değişmemesini sağlayabilirsiniz.

PHP Veri Türleri

- Veri türü, verilerin özelliklerine göre bir kategoride sınıflandırılmasıdır;
 - **Alfanümerik** karakterler dize olarak sınıflandırılır
 - **Tam sayılar** sınıflandırılmış tam sayılardır
 - **Ondalık noktalı** sayılar kayan nokta olarak sınıflandırılır.
 - **Doğru** veya **yanlış** değerler Boolean olarak sınıflandırılır.
 - **Diziler** – bir değişken içinde çok sayıda farklı değer saklayan koleksiyonlarıdır.
- PHP'de açık tanımlanmış veri türleri yoktur. PHP, atanan verilerin niteliklerini analiz ederek veri türlerini belirler. PHP, aşağıdaki veri türlerini dolaylı olarak destekler
- Tamsayı – tam sayılar, örneğin -3, 0, 69. Bir tamsayının maksimum değeri platforma bağlıdır. 32 bitlik bir makinede genellikle 2 milyar civarındadır. 64 bit makineler genellikle daha büyük değerlere sahiptir. PHP_INT_MAX sabiti maksimum değeri belirlemek için kullanılır.

Tamsayılar

- 1453 gibi **ondalık noktası olmayan** tam sayılardır.
 - En basit türlerdir. Hem pozitif hem de negatif basit tam sayılara karşılık gelirler.
 - Tamsayılar, değişkenlere atanabilir veya aşağıdaki gibi ifadelerde kullanılabilir:
-
- `$int_var = 12345;`
 - `$int_negatif_var = -25;`
 - `$another_int = -12345 + 12345;`

Ondalık noktalı veri türleri

- Kayan noktalı sayı, ondalık noktalı bir sayıdır. Tamsayıdan farklı olarak, negatif veya pozitif bir işaret de dahil olmak üzere kesirli veya ondalık noktalı sayıları tutabilir.

```
1. <?php
2.     $n1 = 19.34;
3.     $n2 = 54.472;
4.     $toplam = $n1 + $n2 ;
5.     echo "Ondalıkli sayilarin toplanmasi: " . $toplam ;
6. ?>
```

PHP Boolean(mantıksal) veri türü

- Booleanlar bir anahtar gibidir, 1(doğru) veya 0(yanlış) olmak üzere yalnızca iki olası değeri vardır .
- `$gno=1.5;`
- `$sonuc = ($gno>2);`
- `if ($sonuc)`
- `echo "sonuc : Geçti";`
- `else`
- `echo "Sonuc : Kaldı";`

PHP Dize(string/metin) veri türü

- Dizeler, her karakterin bir bayt ile aynı olduğu karakter dizileridir.
- Bir dize harfleri, sayıları ve özel karakterleri içerebilir ve 2 GB'a kadar (maksimum 2147483647 bayt) olabilir. Bir dize belirtmenin en basit yolu onu tek tırnak içine almaktır (örneğin 'Merhaba dünya!'), ancak çift tırnak işareti de kullanabilirsiniz ("Merhaba dünya!").
- `$a = 'Merhaba Dünya!';`
- `echo $a;`
- `echo "
";`
- `$b = "Merhaba Dünya!";`
- `echo $b;`
- `echo "
";`
- `$siir = 'Kütahya\'nın Pınarları';`
- `echo $siir;`

String yazdırma Alıştırmaları

- Bir dize, bir harf, sayı, özel karakter ve aritmetik değer dizisi veya hepsinin birleşimidir. Bir dize oluşturmanın en basit yolu, dize değişmez değerini (yani dize karakterlerini) aşağıdaki gibi tek tırnak (') içine almaktır:
- `$my_string = 'Merhaba Dünya';`
- Kaçış dizisi değiştirmeleri şunlardır:
 - `\n` yeni satır karakteri ile değiştirilir
 - `\r` satır başı karakteri ile değiştirilir
 - `\t` sekme karakteri ile değiştirilir
 - `\$` dolar işaretinin kendisi ile değiştirilir (\$)
 - `\"` tek bir çift tırnak (") ile değiştirilir
 - `\\` tek bir ters eğik çizgi (\) ile değiştirilir

String yada değişken yazdırma Örneği

- `$my_str = 'Dünya';`
- `echo "Merhaba, $my_str!
";`
- `// browser çıktısı: Merhaba, Dünya!`
- `echo 'Merhaba, $my_str!
';`
- `// browser çıktısı: Merhaba, $my_str!`
- `echo 'Merhaba\tDünya!';`
- `// browser çıktısı: Merhaba\tDünya!`
- `echo "<p>Merhaba\tDünya!</p>";`
- `// browser çıktısı: Merhaba Dünya!`
- `echo 'Kütahya\'nın Pınarları';`
- `// browser çıktısı: Kütahya'nın Pınarları`

Aritmetik operatörler

- Aritmetik operatörler, sayısal veriler üzerinde aritmetik işlemler gerçekleştirmek için kullanılır. PHP aşağıdaki operatörleri destekler.

Şebeke	İsim	Açıklama	Örnek	Çıktı
+	toplama	x ve y'nin toplamı	1 + 1;	2
-	Çıkarma	x ve y arasındaki fark	1 – 1;	0
*	Çarpma işlemi	x ve y'yi çarpar	3 * 7;	21
/	Bölme	x ve y'nin bölümü	45 / 5;	9
%	PHP Modülü	x ve y'nin bölümünden kalanı verir	9 % 2	1
-n	olumsuzlama	n'yi negatif bir sayıya çevirir	-(-5);	5
x . y	birleştirme	x ve y'yi bir araya getirir	"web". " prg ";	web prg
**	Üs alma	X'in y kuvvetini bulur	10**2;	100

Aritmetik Operatörlere Örnek 1

- `<?php`
- `$yüzde=0; $maas=0.0;`
- `$ad="Şenol";`
- `$oran = .15;`
- `$maas = 3000;`
- `$yenimaas=0;`
- `$kesintiOrani=0.05;`
- `$netmaas=0;`
- `$hesaplananMaas= $maas+($maas*$oran);`
- `$kesinti = ($hesaplananMaas*$kesintiOrani);`
- `$netMaas = ($hesaplananMaas - $kesinti);`
- `echo "$ad isimli personele ait bilgiler:

<hr>";`
- `echo "Önceki Maaş : ". $maas. "
";`
- `echo "Zam Oranı : ". $oran."%". "
";`
- `echo "Kesinti : ". $kesinti. "
<hr>";`
- `echo "Hesaplanan Maaş : ". $hesaplananMaas. "
<hr>";`
- `echo "Net Maaş : ". $netMaas. "
<hr>";`
- `?>`

Şenol isimli personele ait bilgiler:

Önceki Maaş : 3000
Zam Oranı : 0.15%
Kesinti : 172.5

Hesaplanan Maaş : 3450

Net Maaş : 3277.5

Aritmetik Operatörlere Örnek 2

```
• <?php {
•   if ($_SERVER["REQUEST_METHOD"] == "POST")
•   {
•       $ad = $_POST["txt_adsoyad"];
•       $maas = $_POST["txt_maas"];
•       $oran = $_POST["txt_oran"];
•       $kesintiorani = $_POST["txt_kesintiorani"];
•       $hesaplananMaas = $_POST["txt_maas"] +
•       ($POST["txt_maas"] * $_POST["txt_oran"]);
•       $kesinti = ($hesaplananMaas *
•       $_POST["txt_kesintiorani"]);
•       $netMaas = ($hesaplananMaas - $kesinti);
•   }
• }?>
• <html>
•   <body>   <table>
•   <form id="maashesapla" method="post" action="<?php
•   echo $_SERVER['PHP_SELF'];?>">
•   <tr><td>Ad Soyad :</td>
•   <td><input type="text" name="txt_adsoyad"
•   value="<?=@$ad;?>">
•   </td></tr>
•   <tr><td>Mevcut Maaş : </td>
•   <td><input type="text" name="txt_maas"
•   value="<?=@$maas;?>"></td>
•   </tr>
•   <tr><td>Artış Oranı : </td>
•   <td><input type="number" name="txt_oran" step="0.01"
•   value="<?=@$oran; ?>"></td>
```

```
• </tr>
• <tr><td>Kesinti Oranı : </td>
•   <td><input type="number" name="txt_kesintiorani"
•   step="0.01" value="<?=@$kesintiorani; ?>"></td>
• </tr>
• <tr><td>Hesaplanan Yeni Maaş : </td>
•   <td><input type="number" name="txt_hesaplananMaas"
•   disabled value="<?=$hesaplananMaas; ?>" ></td>
• </tr>
•   <tr><td>Kesinti Tutarı : </td>
•   <td><input type="number" name="txt_kesinti" disabled
•   value="<?=$kesinti; ?>" ></td>
• </tr>
• <tr><td>Yeni Net Maaş : </td>
•   <td><input type="number" name="txt_netmaas" disabled
•   value="<?=$netMaas; ?>" ></td>
• </tr>
•   <tr><td><input type="submit" name = "gonder"
•   value="Hesapla"></td>
•   <td><input type="reset" name = "Temizle"
•   value="Hesapla"></td></tr>
•   <br><br>
• </form>
• </table>
• </body>
• </html>
```

Ad Soyad :	<input type="text" value="şenoldemirci"/>
Mevcut Maaş :	<input type="text" value="3000"/>
Artış Oranı :	<input type="text" value=",10"/>
Kesinti Oranı :	<input type="text" value=",05"/>
Hesaplanan Yeni Maaş :	<input type="text" value="3300"/>
Kesinti Tutarı :	<input type="text" value="165"/>
Yeni Net Maaş :	<input type="text" value="3135"/>
<input type="button" value="Hesapla"/>	<input type="button" value="Reset"/>

Atama Operatörleri

- Atama operatörleri ile değişkenlere **değer atarız**. Temel atama operatörü eşittir (=).
- Tek bir eşittir işareti, istenilen değişkene istenilen değeri atamamızı sağlar.
- Bununla birlikte aritmetik operatörler ile birlikte de kullanılabilir. Bu operatörler birleşik atama operatörleridir. Bunlar bir değişkenin sonuna bir değer eklemekte ya da değişkendeki sonuca bir sayı eklememizi kolaylaştırır. Bu birleşik atama operatörlerini önce tabloda gösterelim:

operatör	İsim	Örnek	Çıktı	Açıklama
=	atama	\$x = 5;	5	5'i x değişkenine aktarır
+=	Toplayarak ekleme	\$x = 2; \$x += 1;	3	1 değerini x değişkenine ekler
-=	Çıkararak ekleme	\$x = 3; \$x -= 2;	1	2 değerini x değişkeninden çıkarır
*=	Çarparak ekleme	\$x = 0; \$x *=9;	0	9 değerini x değişkeni ile çarpar
/=	Bölerek ekleme	\$x = 6; \$x /=3;	2	6 değerini 3'e böler ve sonucu x'e aktarır
%=	Kalanı bularak ekleme	\$x = 3; \$x %= 2;	1	x'in(3) 2'ye bölümünden kalanı x'e aktarır
.=	birleştirmek	" \$x = 'Güzel'; \$x .= ' Günler!';"	Güzel Günler!	X değişkeni ile "Günler" ifadesini birleştirir

Karşılaştırma operatörleri

- Karşılaştırma operatörleri, değerleri ve veri türlerini karşılaştırmak için kullanılır.
- Mesela \$a değişkeninde bulunan bir değer \$b değişkeni ile aynı mı? Ya da daha mı büyük? Gibi karşılaştırmaları bu operatörler ile yapacağız.
- Tüm karşılaştırma işlemlerinin sonucu **true** yada **false** olur.
- Karar yapılarını (if
 - If else
 - If else if
 - Switch case...

Karşılaştırma operatörleri Kullanımı ve açıklamaları

operatör	İsim	Açıklama	Örnek	Çıktı
$X == y$	Eşitlik	x ve y eşitlerse true değerini döndürür	$1 == "1";$	Doğru veya 1
$X === y$	özdeş	Hem değerleri hem de veri türlerini karşılaştırır.	$1 === "1";$	False veya 0. 1 tamsayı ve "1" ise string olduğundan
$X != y$	PHP Eşit değil	değerler eşit değilse true döndürür	$2 != 1;$	Doğru veya 1
$x <> y$	PHP Eşit değil	değerler eşit değilse true döndürür	$2 <> 1;$	Doğru veya 1
$X > y$	daha büyük	x, y'den büyükse true değerini döndürür	$3 > 1;$	Doğru veya 1
$X < y$	Daha küçük	x, y'den küçükse true döndürür	$2 < 1;$	Yanlış veya 0
$X >= y$	Büyük veya eşit	x, y'den büyük veya ona eşitse true döndürür	$1 >= 1$	Doğru veya 1
$X <= y$	Az veya eşit	x, y'den küçük veya ona eşitse true döndürür	$8 <= 6$	Yanlış veya 0

Mantıksal operatörler

- Mantıksal operatörler ile birden fazla karşılaştırmayı değerlendirip, birleştirebiliriz. Mesela bir değişkenin hem 0 dan büyük mü hem de 100 den küçük mü diye bir karşılaştırmasını mantıksal operatörler sayesinde yaparız.
- Mantıksal operatörleri bir tabloda gösterelim:

Operatör	İsim	Kullanılışı	Açıklaması	Örnek	Açıklama
!	değil	!\$a	\$a'nın değeri olumsuz yani FALSE ise TRUE yani doğru döner. Aksini yapar olumsuz mu diye sorar.	!(\$a>10)	A değişkeni 10'dan büyükmü?
&&	Ve	\$a && \$b	\$a ve \$b'nin değeri olumlu ise olumlu yani TRUE döner, ikisinden birisi olumsuz ise yanlış yani FALSE döner.	\$a > 0 && \$a < 100	a değişkeni 0'dan büyük ve 100'den küçükmü?
	Ya da	\$a \$b	\$a'nın değeri ya da \$b'nin değeri olumlu ise olumlu döner. Birisinin ya da ikisinde olumlu olması yeterlidir.	\$a > 0 \$b > 0	a değişkeni veya b değişkeni 0'dan büyükmü

Mantıksal operatörlere Örnek 1

	Örnek 1 (ve koşulu) &&	Örnek 2 (değil koşulu) !	Örnek 3 (veya koşulu)
Kod →	<pre>\$a=10; \$b=-5; \$sonuc= \$a>0 && \$a<100; echo "Karşılaştırma sonucu : "; var_dump(\$sonuc);</pre>	<pre>\$a=10; \$b=-5; \$sonuc= !(\$a>0 && \$a<100); echo "Karşılaştırma sonucu : "; var_dump(\$sonuc);</pre>	<pre>\$a=10; \$b=-5; \$sonuc= (\$a>0 \$a<100); echo "Karşılaştırma sonucu : "; var_dump(\$sonuc);</pre>
Sonuç →	Karşılaştırma sonucu : bool(true)	Karşılaştırma sonucu : bool(false)	Karşılaştırma sonucu : bool(true)

```
$gno=2.5; //genel not ortalaması
$sinif=3; //kaçıncı sınıfta
olduğu
$kredi=15; // aldığı kredi
$sonuc= ($sinif < 3 && $kredi <
15) || ($gno>=3 );
echo "Karşılaştırma sonucu : ";
var_dump($sonuc);
```

Burada 2 koşul vardır. Ancak koşullardan biri 2 durumludur. Koşullardan biri veya ikisi sağlanırsa TRUE olacak.

1.koşul: sınıfın 3'ten küçük ve kredisinin 15'ten küçük olması.

2.koşul: gno'nun 3 veya daha fazla olması.

1. Veya 2. koşul sağlanırsa sonuç değişkeni true olur.

Mantıksal operatörler Örnek 2

- Bilgi: **Artık yıllar 400'e bölünebilir** veya **4'e bölünebilir ama 100'e bölünemez**. Buradan yola çıkarak bir yılın artık yıl olup olmadığını bulalım.

- //bazı artık yıllar: 1980, 1984, 1988, 1992, 1996, 2000, 2004, 2008, 2012, 2016, 2020, 2024 vb

```
$yıl = 2024;  
// Artık yıllar 400'e veya 4'e bölünebilir ama 100'e bölünemez  
if(($yıl % 400 == 0) || (($yıl % 100 != 0) && ($yıl % 4 == 0)))  
{  
    echo "$yıl artık bir yıldır.";  
}  
else  
{  
    echo "$yıl artık yıl DEĞİLDİR!.";  
}
```


arttırma ya da eksiltme operatörleri

- Bir değişkendeki sayısal değeri bir arttırmak için çift artı (++) operatörünü, bir eksiltmek için de çift eksi (- -) operatörünü kullanırız.
- Ancak bunun kullanımına dikkat etmek gerekir.
- Zira \$a++ ile ++\$a farklı sonuçlar verir. Çünkü \$a++ ifadesi önce \$a değişkenini kullan sonra 1 arttır anlamına gelmektedir. ++\$a ifadesi ise önce 1 arttır sonra kullan anlamına gelmektedir.

işlem	Anlamı	Açıklama
++\$x	Önce artır, sonra işlem yap	\$x'i bir artırır, sonra \$x döndürür
\$x++	Önce işlem yap, Sonra artır	\$x döndürür, sonra \$x'i bir artırır
--\$x	Önce eksilt, sonra işlem yap	\$x'i bir azaltır, sonra \$x döndürür
\$x--	Önce işlem yap, Sonra eksilt	\$x döndürür, sonra \$x'i bir azaltır

Artırma ve Eksiltme Operatörlerine Örnek

- `$a=15; $b=6;`
- `echo "
";`
- `echo $a++; // $a'nın değeri 15 yazar ve değeri artırılır.`
- `echo "
";`
- `echo $a; // burada 16 yazar`
- `echo "
";`
- `$a=15; // tekrar 15 değerini atadık.`
- `echo ++$a; // burada $a önce artırılır yani 16 olur. sonra ekrana yazılır. 16`
- `echo "
 <hr>";`
- `$a=15; $b=6;`
- `echo $a--; // $a'nın değeri 15 yazar ve değeri azaltılır.`
- `echo "
";`
- `echo $a; // burada 14 yazar`
- `echo "
";`
- `$a=15; // tekrar 15 değerini atadık.`
- `echo --$a; // burada $a önce azaltılır yani 14 olur. sonra ekrana yazılır. 14`

15
16
16
15
14
14

PHP Koşullu İfadeler

- Çoğu programlama dilinde olduğu gibi PHP, çalışma zamanında mantıksal veya karşılaştırmalı test koşullarının sonuçlarına göre farklı eylemler gerçekleştiren kodlar yazmanıza da olanak tanır.
- Bu, true veya false olarak değerlendirilen ifadeler şeklinde test koşulları oluşturabileceğiniz ve bu sonuçlara dayanarak belirli eylemleri gerçekleştirebileceğiniz anlamına gelir.
- PHP'de karar vermek için kullanabileceğiniz birkaç ifade vardır:
 - if ifadesi
 - if ...else ifadesi
 - if ...elseif...else ifadesi
 - Switch ... case ifadesi

if karar yapısı

- if ifadesi, yalnızca belirtilen koşul doğru olarak değerlendirilirse bir kod bloğunu yürütmek için kullanılır.
- Bu, PHP'nin en basit koşullu ifadeleridir ve şöyle yazılabilir:
- if (koşul)
- {
- // Yürütülecek kod
- }
- Aşağıdaki örnek; geçerli gün Cuma ise "İyi bir hafta sonu geçirin!" şeklinde bir mesaj yazdırmak içindir:

```
<?php
// D parametresi kısa gün adını, l parametresi uzun
gün adını verir.
$d = date("l");
echo "Bugün günlerden :$d <br>";
if($d == "Fri")
{
    echo " İyi bir hafta sonu geçirin!";
}
?>
```

if...else yapısı

- *if ifadesine başka* bir ifade ekleyerek alternatif bir seçim sağlayarak karar verme sürecini iyileştirebilirsiniz.
- *if...else* ifadesi , belirtilen koşul doğru olarak değerlendirilirse bir kod bloğunu ve yanlış olarak değerlendirilirse başka bir kod bloğunu yürütmenize izin verir. Şu şekilde yazılabilir:

```
if (koşul){  
    // Koşul doğruysa yürütülecek  
kod } else {  
    // Koşul yanlışsa yürütülecek kod  
}
```

If else Örneği.

- // D parametresi kısa gün adını, l parametresi uzun gün adını verir.
- `$d = date("l");`
- `echo "Bugün günlerden :$d
";`
- `if($d == "Fri")`
 - `{`
 - `echo " İyi bir hafta sonu geçirin!";`
 - `}`
 - `else{`
 - `echo "İyi Günler!";`
 - `}`

Gün adı Cuma ise "İyi bir hafta sonu geçirin!" yazısını değilse "İyi Günler!" yazısını çıktıda görürüz.

if else Örnek 2.

- Aşağıdaki örnekte kullanıcı adı ve şifre doğrulaması yapılmaktadır. Tanımlanan `kullanici_adi` ve `$sifre` değişkenlerinin değeri karşılaştırılan kullanıcı adı ve şifre ile aynı ise if bloğu, farklı ise else bloğu çalışacaktır.

```
$kullanici_adi="admin";
$sifre="1234";
if ($kullanici_adi=="admin" && $sifre=="1234" )
{
    echo "Hoşgeldin $kullanici_adi";
}
else
{
    echo "sistemde $kullanici_adi adında bir kullanıcı yok ya da
yanlış şifre girdiniz.</br>";
}
```

if else Örnek 3.

- Şimdiki örneğimizde ise tanımlı olan sayı
 - *tek* ise sayının *küpünü*,
 - *çift* ise *karesini*
- hesaplayıp ekranda yazdıralım. Sayının tek yada çift olduğunu bulmak için mod operatörünü kullanacağız.
- *Sayının 2'ye bölümünden kalan 0 ise çift, değilse ise tek sayıdır.*
- `$sayi=5;`
- `if($sayi % 2==0) //sayının modu bulunuyor.`
- `echo "Sayı çift";`
- `Else`
- `echo "Sayı tek";`

The Ternary Operator (üçlü koşul operatörü)

- Üçlü operatör, if...else deyimlerini yazmanın kısa bir yolunu sağlar.
- Üçlü operatör, soru işareti (**?** ve **:**) sembolü ile temsil edilir ve üç işlenen alır.
 - kontrol edilecek bir koşul, (***mantıksal koşul***)
 - Doğru sonuç için true ve (**?**)
 - Yanlış sonuç için bir false. (**:**)
- Yazım Şekli aşağıdaki gibidir.
 - *(Koşul) ? (koşul doğruysa bu kod) : (koşul yanlışsa bu kod);*
- Örnek için sonraki slayda bakınız.



Ternary operatörü Örnek

if else ile → 18 yaşından küçükse "Çocuk" değilse "Yetişkin yazacak"

```
if($yaş < 18)
    echo 'Çocuk';
else {
    echo 'Yetişkin';
}
```

Ternary ile → 18 yaşından küçükse "Çocuk" değilse "Yetişkin yazacak"

```
echo ($age < 18) ? 'Çocuk' : 'Yetişkin';
//Görüldüğü gibi kodlar yukarıdaki if else bloğuna göre hayli azaldı.
```

Ternary ile farklı bir yazım şekli

```
$username="admin";
$password="1234";
$is_user_logged_in = ($username=="admin" && $password=="1234");
$result = $is_user_logged_in
? 'Bağlandı'
: 'Bağlanamadı';
echo $result;
```

Kullanıcı adı ve Şifre doğru ise "Bağlandı",
yanlışsa "Bağlanmadı" yazar.

if...elseif...else yapısı

- Birden çok *if...else* ifadesini birleştirmek için kullanılan özel bir ifadedir. Bu nedenle, bu ifadeyi kullanarak birden çok koşulu kontrol edebiliriz.
- Yürütme üstten başlar ve her bir if koşulu için kontrol edilir.
- Doğru olduğu değerlendirilen if bloğunun ifadesi yürütülür.
- if koşulunun hiçbirisi doğru değilse, son else bloğu yürütülür.

```
if(koşul1)
{
    //koşul1 doğruysa yürütülecek kod
}
else if(koşul2)
{
    //koşul2 doğruysa yürütülecek kod
}
else if(koşul3)
{
    //koşul3 doğruysa yürütülecek kod
}
...
else
{
    //tüm koşullar yanlışsa yürütülecek kod
}
```

if else if Örnek 1

- `$i` değişken değeri `10` ile kıyaslanıyor, `büyük` olma, `küçük` olma ve `eşit` olma gibi `3` ihtimal var. Bu yüzden `3` ihtimali değerlendirmek için `if else if` yapısını kullanacağız. `2` ihtimal olsaydı `if else` yeterli olacaktı.
- `$i = 20;`
- `if ($i > 10)`
- `echo "i büyük";`
- `else if ($i < 10)`
- `echo "i küçük";`
- `else if ($i == 10)`
- `echo "i eşit";`
- `else`
- `echo "i mevcut değil";`

Örnek 2

- Bir başka örnek daha: Şimdiki örneğimiz ise sıcaklık değerine göre maddenin *sıvı*, *gaz* yada *katı* halde olduğunu bulmayı sağlayacağız.
- `$sıcaklık = -10;`
- `if($sıcaklık >= 100)`
- `echo "Madde $sıcaklık derecede gaz halindedir.";`
- `else if ($sıcaklık > 0)`
- `echo "Madde $sıcaklık derecede sıvı halindedir.";`
- `else`
- `echo "Madde $sıcaklık derecede katı halindedir.";`
-

if else if Örnek 3

- Bu örneğimizde kişinin vücut kitle endeksini bulup durumunu yazdıracağız.
- $VKİ = \text{Vücut kitle endeksi}$.
- $VKİ$, Ağırlığınızı boyunuzun karesine böldüğünüzde (kg/m^2) çıkan sonuçtur. Bu sonuç, zayıf, kilolu, fazla kilolu ya da obez olup olmadığını belirtir. Vücut kitle endeksi ($VKİ$) olarak kısaltırsak;
- $VKİ$ 18 ile 25 aralığındaysa normal,
- $VKİ$ 25 ile 30 aralığındaysa kilolu,
- $VKİ$ 30 ve daha yüksekse obez,
- $VKİ$ 35 ve daha fazlaysa ciddi obez olarak kabul edilir.
- Buna göre kodlarımızı yazalım.

if else if Örnek 3 Çözümü

```
1. $boy = 1.80;
2. $kilo = 88;
3. $sonuc = "";
4. $vki = $kilo / ($boy * $boy) ;
5. if ($ vki < 18)
6.     $sonuc = "zayıf";
7. else if ($ vki < 25)
8.     $sonuc = "normal";
9. else if ($ vki < 30)
10.    $sonuc = "kilolu";
11.else
12.    $sonuc = "obez";
13.echo "sonuç : ".$sonuc;
```

Bu örnekte 4. Satırda oran(VKİ) hesaplanıp sırasıyla 5, 7 veya 9. Satırlardaki if koşulları değerlendirilir. Bunlardan birisine koşul uyarsa oradaki deyimler çalışır. Koşul yani oran değişkeni hiç birisine uymazsa else ifadesindeki deyim çalışır. Eğer oran değişkeni bir koşula uyarsa diğer koşullara hiç bakılmaz.

if else if Örnek 4

- Bu örnekte basit bir vergi hesabı programı geliştirmek istiyoruz. Gelir vergisinin yandaki kurallara göre belirlendiğini varsayalım.

Gelir	Vergi Oranı	Vergi Tutarı
≤ 150,000	%25	?
≤ 300,000	%30	?
≤ 600,000	%35	?
> 600,000	%40	?

```
$vergi=0;
$gelir = 170000;
if ($gelir <= 150000)
    $vergi = $gelir * 0.25;
else if ($gelir <= 300000)
    $vergi = $gelir * 0.30;
else if ($gelir <= 600000)
    $vergi = $gelir * 0.35;
else
    $vergi = $gelir * 0.40;
echo "vergi Tutarı :".$vergi."<br>";
echo "Net Kazanç :".($gelir - $vergi);
```


Switch Case Yapısı

- **Switch-case** ifadesi, hemen hemen aynı şeyi yapan **if-else if-else** ifadesinin bir alternatifidir. Switch-case ifadesi, bir eşleşme bulana kadar bir değişkeni bir dizi değere karşı test eder ve ardından bu eşleşmeye karşılık gelen kod bloğunu yürütür.
- **switch**(degisken){
 - **case** value1:
 - //yürütülecek kod
 - **break**;
 - **case** value2:
 - //yürütülecek kod
 - **break**;
 -
 - **default**:
 - //yürütülecek kod (tüm durumlar eşleşmezse);
 - }

Not: Her **case** bloğunda mutlaka **break** kullanılmalıdır.

Switch case Örnek 1

- `$mevsim="sonbahar";`
- `echo "$mevsim ayları : ";`
- `switch ($mevsim){`
- `case "ilkbahar" : echo "mart nisan mayıs";break;`
- `case "yaz" : echo "haziran temmuz ağustos";break;`
- `case "sonbahar" : echo "eylül ekim kasım";break;`
- `case "kış" : echo "aralık ocak şubat";break;`
- `default:echo "yanlış mevsim";break;`
- `}`
- Not: `$mevsim` değişken değeri hangi `case` satırına eşleşiyorsa oradaki kodlar çalışır. Bulduğu koşuldan çıkmak için mutlaka `break`; kullanılmalıdır. Hiçbir koşul ile eşleşmez ise `default` satırı çalışır.

Switch case Örnek 2

- 100 üzerinden tanımlanmış bir notun 5 dereceye göre karşılığını bulalım.
- `$notu=79;`
- `switch ($notu){`
- `case ($notu<=100 && $notu>=90): echo "5-Pekiyi"; break;`
- `case ($notu<=89 && $notu>=80): echo "4-İyi"; break;`
- `case ($notu<=79 && $notu>=70): echo "3-Orta"; break;`
- `case ($notu<=69 && $notu>=50): echo "2-Geçer"; break;`
- `case ($notu>=0 && $notu < 50): echo "1-Başarısız"; break;`
- `default: echo "Geçersiz Not";`
- `}`

`$notu` değişken değeri 79, 3. case satırındaki sayı aralıklarına uygun düştüğü için ekranda "3-Orta" yazısını göreceğiz. Kalan diğer koşulları incelemeyecektir.

`$notu` değişken değeri 110 veya -5 gibi bir sayı olsaydı "Geçersiz Not" yazacaktır.

Dizi Nedir?

- PHP'de, bir dizi birden çok öğeyi tek bir değişkende depolamaya olanak sağlayan bir veri yapısıdır.
- Bu öğeler anahtar-değer çiftleri olarak depolanır. Aslında, bir öğe listesini depolamak gerektiğinde dizi kullanabilirsiniz.
- Çoğunlukla, dizideki tüm öğeler benzer veri türlerine sahip.
- Örneğin, meyve adları saklamak istediğinizi varsayalım. Bir dizi olmadan, farklı meyve adlarını depolamak için birden çok değişken oluşturma yoluna gidecektiniz.
- Öte yandan, meyve adlarını saklamak için bir dizi kullanırsanız, aşağıdaki gibi görünecektir:
- `<?php`
- `$array_fruits = array('Apple', 'Orange', 'Watermelon', 'Mango');`
- `?>`
- Gördüğünüz gibi \$array_fruits değişkenini farklı meyve adlarını saklamak için kullandım. Bu yaklaşımın en iyi yanı \$array_fruits dizi değişkenine daha sonra istediğiniz kadar eleman ekleyebilmenizdir.

Bir Dizi Nasıl Başlatılır

- Bu bölümde, bir dizi değişkeni nasıl başlatılır ve bu değişkene nasıl değer eklenir, bunu ele alacağız.
- Bu diziyi başlatmak için, birkaç farklı yolu vardır. Çoğu durumda, `array()` bir diziyi başlatmak için kullanılan dil yapısıdır.

```
<?php
```

```
$array = array();
```

```
?>
```

- Yukarıdaki kod parçasında `$array` değişkeni boş bir dizi olarak başlatılır.
- Bir diziyi başlatmak için aşağıdaki sözdizimini de kullanabilirsiniz.

```
<?php
```

```
$array = [];
```

```
?>
```

bir diziye nasıl öge eklenir.

```
<?php
$array = [];
$array[] = 'One';
$array[] = 'Two';
$array[] = 'Three';
echo '<pre>';
print_r($array);
?>
```

```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
)
```

- Yukarıdaki kod parçasının yandaki çıktıyı üretmesi gerekir.
- Burada unutulmaması gereken önemli şey bir dizi indeksi 0 ile başlar.
- Bir indeks belirtmeden bir dizi için yeni bir öge eklediğinizde, dizinin indeksi otomatik olarak atanır.
- Kuşkusuz, değerleri olan bir dizi de oluşturabilirsiniz.
- Hangi değerlerin olacağını biliyorsanız bir diziyi bildirmek için en kısa yol aşağıdadır.

```
<?php
$array = ['One', 'Two', 'Three'];
?>
```

Dizi Öğelerine Nasıl Erişilir

- Önceki bölümde, bir dizi değişkeninin nasıl başlatıldığını tartıştık. Bu bölümde, dizi öğelerine erişmenin birkaç farklı yolunu ele alacağız.
- Dizi öğelerine erişmenin açıkça görünen ilk yolu, öğeleri dizi anahtarı veya indeks kullanarak almaktır.

```
<?php ' ] ;
```

```
$array = [ 'One', 'Two', 'Three  
// dizinin ilk elemanına erişim  
echo $array[0];  
echo "<br>";  
  
// dizinin ikinci elemanına erişim  
echo $array[1];  
echo "<br>";  
  
// dizinin üçüncü elemanına erişim  
echo $array[2];  
echo "<br>";  
?>
```

Döngü ile dizi elemanlarına erişim

- Önceki slayttaki kodu yazmanın temiz yolu, dizinin öğeleri üzerinde yineleme yapmak için **foreach** döngüsü kullanmaktır.

```
<?php
$array = ['One', 'Two', 'Three'];

foreach ($array as $element) {
    echo $element;
    echo '<br>';
}
?>
```

- Benzer şekilde, dizi öğeleri üzerinde dolaşmak için **for** döngüsünü de kullanabilirsiniz.

```
<?php
$array = ['One', 'Two', 'Three'];
$array_length = count($array);

for ($i = 0; $i < $array_length; ++$i)
{
    echo $array[$i];
    echo '<br>';
}
?>
```


PHP'deki Dizi Türleri

- Bu bölümde, PHP'de kullanabileceğiniz farklı türdeki dizileri tartışacağız.

Sayısal İndeksli Diziler

- Sayısal indeks içeren bir dizi indekslenmiş dizi kategorisinde yer alır. Aslında, bu makalede şimdiye kadar tartıştığımız örnekler indekslenmiş dizilerdir.
- Açıkça belirtmediğiniz zaman sayısal indeks otomatik olarak atanır.

```
<?php
$array = ['One', 'Two', 'Three'];
?>
```

- Yandaki örnekte, her öğe için bir indeks açıkça belirtmedik, bu yüzden sayısal indeks otomatik olarak başlatılacaktır.
- Tabii ki, bir diziyi sayısal indeks kullanarak aşağıdaki kod parçasında gösterildiği gibi oluşturabilirsiniz.

```
<?php
$array = [];
$array[0] = 'One';
$array[1] = 'Two';
$array[2] = 'Three';
?>
```

PHP'deki Dizi Türleri

- Bu bölümde, PHP'de kullanabileceğiniz farklı türdeki dizileri tartışacağız.

İlişkili (Associative) Diziler

- İlişkili dizi indekslenmiş diziye benzer, ancak dizi anahtarları için karakter dizisi değerlerini kullanabilirsiniz.
- Bir ilişkisel dizi nasıl tanımlanır, hadi görelim.

```
<?php
$employee = [
    'name' => 'John',
    'email' => 'john@example.com',
    'phone' => '1234567890',
];
```

- Yandaki örneğe Alternatif olarak, aşağıdaki sözdizimini de kullanabilirsiniz.

```
<?php
$employee = [];
$employee['name'] = 'John';
$employee['email'] = 'john@example.com';
$employee['phone'] = '1234567890';
?>
```

İlişkisel dizi oluşturma

```
/* İlişkisel bir dizi oluşturmak için ilk yöntem. */
$student_one = array("Matematik"=>95, "Fizik"=>90,
                    "Kimya"=>96, "İngilizce"=>93,
                    "Bilgisayar"=>98);

/* İlişkisel bir dizi oluşturmak için ikinci yöntem. */
$student_two["Matematik"] = 95;
$student_two["Fizik"] = 90;
$student_two["Kimya"] = 96;
$student_two["İngilizce"] = 93;
$student_two[" Bilgisayar "] = 98;

/* Öğelere doğrudan erişim */
echo "Bu öğrenci için notlar:\n";
echo "Matematik:" . $student_two["Matematik"], "\n";
echo "Fizik:" . $student_two["Fizik"], "\n";
echo "Kimya:" . $student_two["Kimya"], "\n";
echo "İngilizce:" . $student_one["İngilizce"], "\n";
echo " Bilgisayar :" . $student_one[" Bilgisayar "], "\n";
?>
```

İlişkili dizilerdeki elemanlara erişim

- Bir ilişkisel dizinin değerlerine erişmek için indeks veya foreach döngüsü kullanabilirsiniz.

```
<?php
$employee = [
    'name' => 'John',
    'email' => 'john@example.com',
    'phone' => '1234567890',
];

// dizideki isim(name) elemanına erişim
echo $employee['name'];

// tüm elemanlara erişim
foreach ($employee as $key => $value) {
    echo $key . ':' . $value;
    echo '<br>';
}
?>
```

Gördüğünüz gibi burada doğrudan sorgulayarak adı aldık ve sonra dizideki tüm anahtar-değer çiftlerini almak için **foreach** döngüsünü kullandık.

- Diziye birden fazla eleman ekleme(array_push)
 - `array_push($dizi,"kemal","canan","veli");`
- diziden eleman silme. iki şekilde olur: 1-Dizi değerine göre;örneğin "ahmet". 2- Dizi indis noya göre; örneğin 2 nolu indis.
 - `$dizi = array_diff($dizi, array('kemal')) ; //dizi değerine göre silme işlemi`
- eğer dizi elemanlarının indis değerlerini düzenlemek istiyorsak;
 - `array_values($dizi);`
- işlemi yaparız.
- diziden eleman silmenin bir diğer yolu ise indis değerine göre işlem yapmaktır. Aşağıdaki kodlar dizi indis değerine göre silme işlemi yapar.
- Silmek istediğimiz dizi elemanını biliyorsak ki örneğin 3. eleman;
 - `unset($dizi[2]); // 3.eleman dedik ancak 2 yazdık. Unutmayın dizi indisleri sıfırdan başlar.`

Anahtar değerli dizileri foreach ile yazdırma

- `$ogr=array(array("ad"=>"şenol","soyad"=>"demirci","harfnot"=>"AA"),`
- `array("ad"=>"Şahin","soyad"=>"Sağlam","harfnot"=>"BB")`
- `);`
- `$ogr[2]["ad"]="ali";`
- `$ogr[2]["soyad"]="veli";`
- `$ogr[2]["harfnot"]="CC";`
- `echo "<hr>";`
- `foreach($ogr as $deger)`
- `{`
- `echo "Ad: ".$deger["ad"]."
";`
- `echo "SoyAd: ".$deger["soyad"]."
";`
- `echo "harfnot: ".$deger["harfnot"]."
";`
- `echo "<hr>";`
- `}`

Diziye sonradan bir eleman ekledik

Ad: şenol
SoyAd: demirci
harfnot: AA

Ad: Şahin
SoyAd: Sağlam
harfnot: BB

Ad: ali
SoyAd: veli
harfnot: CC

Foreach döngüsünün sonucu

Dizilerde arama işlemi(in_array ve array_search)

- in_array fonksiyonu aranan bir değeri dizide bulursa TRUE bulamazsa FALSE döndürür.
- Array_search fonksiyonu ise; in_array gibi çalışır, aranan değer dizide bulunursa TRUE yerine, bulunan elemanın ANAHTARINI döndürür, bulamazsa yine FALSE döndürür.
- \$aylar=array('ocak','mart','nisan','mayıs');
- if(in_array('nisan',\$aylar)
 - Echo 'bulundu';
 - Else
 - Echo 'bulunamadı';
- //Aynı arama işlemi array_search ile yapacak olursak;
- \$key=array_search('nisan',\$aylar);
- If(!empty(\$key)
 - Echo 'aradığınız değer bulundu ve anahtar değeri :'.\$key;
 - Else
 - Echo 'aradığınız değer bulunamadı';

Çok Boyutlu Diziler

- Şimdiye kadar tartıştığımız örneklerde, dizi ögesi olarak skalar değerleri kullandık. Aslında, dizi ögesi olarak diğer dizileri bile saklayabilirsiniz — bu çok boyutlu bir dizidir.
- Haydi bir örneğe bakalım.

```
<?php
$employee =
[
    'name'      => 'John',
    'email'     => 'john@example.com',
    'phone'     => '1234567890',
    'hobbies'   => ['Football', 'Tennis'],
    'profiles'  => ['facebook' => 'johnfacebook', 'twitter' => 'johntwitter']
];
?>
```

- Gördüğünüz gibi, \$employee dizisinin hobbies anahtarı hobiler dizisini tutar. Aynı şekilde, profiles anahtarı bir ilişkisel dizi ile farklı profilleri tutar.

Çok boyutlu diziler

- Bu dizi sütun ve satırlardan oluşan bir tablo olarak düşünülebilir. Bu tabloda, ilk belirtilen array kelimesi içerisindeki her bir yeni array sütunları, ve içlerindeki her bir değer de o sütunlara ait satırları temsil etmektedir.
- Kısacası çok boyutlu diziler; dizi içinde dizi olarak ifade edilir.
- Dizinin anahtarları tanımlama sırasında belirtilmediği için php tarafından sıfırdan başlayarak otomatik oluşturulur.

Örneğin öğrenci adlarını tutacağımız bir dizi değişkene ihtiyacımız olduğunda;

```
$ogrenciler=array("ali", "veli", "Ahmet");
```

Şeklinde bir tanımlama yapabiliyorduk. Fakat bu öğrencilerin başka bilgilerini de saklamak istersek ki örneğin vize notu, final notu ve harf_notu gibi. İşte bu durumda dizi içinde dizi tanımlamak gerekir. Yandaki şekli inceleyiniz.

Ad soyad	Vize	Final	Harf_notu
Ali	34	55	DD
Veli	56	87	CB
Ahmet	80	90	BB

Çok boyutlu diziler 2

- Önceki slayttaki örneğimizi iki boyutlu dizi şeklinde yazalım

```
$ogrenciler=array(  
    array("ali",34,55, "DD"),  
    array("veli",56,87, "DD"),  
    array("ahmet",80,90, "DD")  
);
```

```
Array  
(  
    [0] => Array  
        (  
            [0] => ali  
            [1] => 34  
            [2] => 55  
            [3] => DD  
        )  
)
```

Not: gördüğünüz gibi dizi içinde 3 tane array(dizi) yazdık. Her bir Sonuna virgül konur. Ancak son dizide virgül yok!!!

Dikkat ettiyseniz dizi elemanlarına anahtar kullanmadık. Bu yüzden Hangi veri neyi ifade ediyor anlaşılmıyor. Bu dizimizi anahtar Kullanarak tanımlayacak olursak daha anlaşılır olacaktır. Şöyleki;

```
$ogrenciler=array(  
    array("ad"=>"ali","vize"=>34,"final"=>55, "harf_not"=>"DD"),  
    array("ad"=>"veli","vize"=>56,"final"=>87, "harf_not"=>"DD"),  
    array("ad"=>"ahmet","vize"=>80,"final"=>90, "harf_not"=>"DD")  
);
```

```
Array  
(  
    [0] => Array  
        (  
            [ad] => ali  
            [vize] => 34  
            [final] => 55  
            [harf_not] => DD  
        )  
)
```

Çok boyutlu diziyi yazdıralım

- ---for döngüsü ile yazdırma---

- \$ogrenciler=array(array("ad"=>"ali","vize"=>34,"final"=>55,"harfnot"=>"DD"),
- array("ad"=>"veli","vize"=>56,"final"=>87,"harfnot"=>"DD"),
- array("ad"=>"ahmet","vize"=>80,"final"=>90,"harfnot"=>"DD")
-);
- for (\$i = 0; \$i < count(\$ogrenciler); \$i++) {
- echo \$ogrenciler[\$i]["ad"]."
";
- echo \$ogrenciler[\$i]["vize"]."
";
- echo \$ogrenciler[\$i]["final"]."
";
- echo \$ogrenciler[\$i]["harfnot"]."
";
-
- echo "<hr>";
- }

- ---foreach döngüsü ile yazdırma---

- \$ogrenciler=array(array("ad"=>"ali","vize"=>34,"final"=>55,"harfnot"=>"DD"),
- array("ad"=>"veli","vize"=>56,"final"=>87,"harfnot"=>"DD"),
- array("ad"=>"ahmet","vize"=>80,"final"=>90,"harfnot"=>"DD"));
- foreach (\$ogrenciler as \$key => \$value) {
- echo \$value["ad"]."
";
- echo \$value["vize"]."
";
- echo \$value["final"]."
";
- echo \$value["harfnot"]."
"; echo "<hr>";
- }
- Yada!!!!!!!!!!!!!!!!!!!!!!
- foreach (\$ogrenciler as \$key => \$value) {
- foreach (\$value as \$key => \$value) {
- echo \$key."=>".\$value."
";
- }
- echo "<hr>";
- }

Çok boyutlu diziler(araba örneği)

Adı	adet	satılan
Volvo	22	18
Bmv	15	13
Saab	5	2
Land rover	17	15

Yandaki gibi bir tabloyu dizi olarak tanımlayacak olursak;

```
$cars = array  
(  
    array("Volvo",22,18),  
    array("BMW",15,13),  
    array("Saab",5,2),  
    array("Land Rover",17,15)  
);
```

Dizi değişkenlerini görmek için;

```
<?php  
echo $cars[0][0].": Mevcut: ".$cars[0][1].", Satılan: ".$cars[0][2]."<br>";  
echo $cars[1][0].": Mevcut : ".$cars[1][1].", Satılan : ".$cars[1][2]."<br>";  
echo $cars[2][0].": Mevcut : ".$cars[2][1].", Satılan : ".$cars[2][2]."<br>";  
echo $cars[3][0].": Mevcut : ".$cars[3][1].", Satılan : ".$cars[3][2]."<br>";  
?>
```

```
<?php  
for ($row = 0; $row < 4; $row++) {  
    echo "<p><b>Araba: $row</b></p>";  
    echo "<ul>";  
    for ($col = 0; $col < 3; $col++) {  
        echo "<li>".$cars[$row][$col]."</li>";  
    }  
    echo "</ul>";  
}  
?>
```

Çok boyutlu diziler(araba örneği) key yöntemi

```
1. $cars2 = array(
2.     array("marka"=> "Volvo", "adet"=>22,"satılan"=>18),
3.     array("marka"=> "BMW", "adet"=>15,"satılan"=>13),
4.     array("marka"=> "Saab", "adet"=>5,"satılan"=>2),
5.     array("marka"=> "Land Rover", "adet"=>17,"satılan"=>15)
6. );
7. foreach ($cars2 as $key => $value) {
8.     echo $value["marka"]."--".$value["adet"]."--".$value["satılan"]. "<br>";
9. }
10. //yada aşağıdaki gibi iç içe foreach döngüsü oluşturup yazdırabiliriz.
11. foreach ($cars2 as $key => $value) {
12.     echo "<ul><b><u>".$value["marka"]."</u></b>";
13.     foreach ($value as $key=> $value) {
14.         echo "<li>".$key ."--".$value ."</li>";
15.     }
16.     echo "</ul> ";
17. }
```

Volvo--22--18
BMW--15--13
Saab--5--2
Land Rover--17--15

Ekran çıktısı

Volvo

- marka--Volvo
- adet--22
- satılan--18

BMW

- marka--BMW
- adet--15
- satılan--13

Saab

- marka--Saab
- adet--5
- satılan--2

Land Rover

- marka--Land Rover
- adet--17
- satılan--15

Çok boyutlu diziler – örnekler...

- `$Ornek=array('istanbul'=>array('sirinevler', 'bahcelievler', 'bakirköy'),
 'ankara'=>array('kızılay', 'yenimahalle', 'çankaya'))`;

```
foreach($Ornek as $Anahtar=>$deger){ //iki kez dönecek
```

```
    echo "$Anahtar<hr/>";
```

```
        foreach($deger as $oge){ // her birisinin eleman sayısı kadar  
dönecek
```

```
            echo "$Or<br/>";
```

```
        }
```

```
    }
```

İstanbul	şirinevler	bahçelievler	bakırköy
Ankara	Kızılay	yenimahalle	çankaya

Çok boyutlu bir dizinin değerlerine erişim

```
<?php
$employee = [
    'name' => 'John',
    'email' => 'john@example.com',
    'phone' => '1234567890',
    'hobbies' => ['Football', 'Tennis'],
    'profiles' => ['facebook' => 'johnfb', 'twitter' => 'johntw']
];

// access hobbies
echo $employee['hobbies'][0];
// Football

echo $employee['hobbies'][1];
// Tennis

// access profiles
echo $employee['profiles']['facebook'];
// johnfb

echo $employee['profiles']['twitter'];
// johntw
?>
```

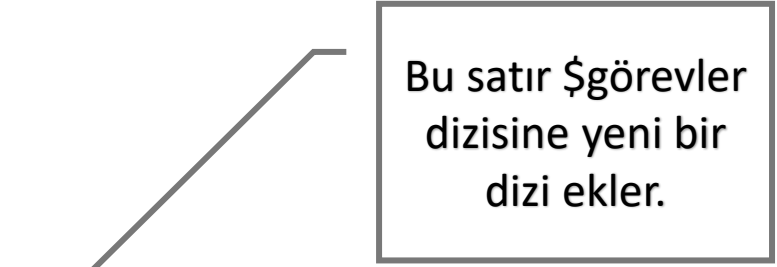
- Gördüğümüz gibi çok boyutlu bir dizinin öğelerine indeks veya anahtar ile her dizi bölümüne erişilebilir.

Çok boyutlu diziler

- Aşağıdaki örnek, iki boyutlu bir dizi tanımlamak için bir array dizisi kullanır:
- `$gorevler = [`
- `['php programlama öğrenimi', 2],`
- `['php pratikleri', 2],`
- `['extra Çalışma', 8],`
- `['Eksersiz yapma' 1],`
- `];`
- Bu `$gorevler` dizisinde, ilk boyut görevleri temsil eder ve ikinci boyut her bir görev için harcanan saati belirtir.
- Çok boyutlu bir dizideki tüm öğeleri görüntülemek için `print_r()` işlevi şöyle kullanırsınız: `print_r($_gorevler);`

PHP çok boyutlu diziye öge ekleme

- Çok boyutlu bir diziye öge eklemek için aşağıdaki sözdizimini kullanırsınız:
- `$array[] = [element1, element2, ...];`
- Örneğin, `$gorevler` dizinin sonuna bir öge eklemek için aşağıdakileri kullanırsınız:
- `<?php`
- `$gorevler = [`
- `['php programlama öğrenimi', 2],`
- `['php pratikleri', 2],`
- `['extra Çalışma', 8],`
- `['Eksersiz yapma', 1],`
- `];`
- `$gorevler[] = ['PHP de önemli bir şeyler yazın', 2];`
- `print_r($gorevler);`
- `?>`



Bu satır `$gorevler` dizisine yeni bir dizi ekler.

PHP çok boyutlu diziden öğe kaldırma (unset)

Dizinin Önceki hali

```
Array
(
    [0] => Array
        (
            [0] => php programlama öğrenimi
            [1] => 2
        )
    [1] => Array
        (
            [0] => php pratikleri
            [1] => 2
        )
    [2] => Array
        (
            [0] => extra çalışma
            [1] => 8
        )
    [3] => Array
        (
            [0] => Eksersiz yapma
            [1] => 1
        )
    [4] => Array
        (
            [0] => PHP de önemli bir şeyler yazın
            [1] => 2
        )
)
```

Dizinin Sonraki hali

```
Array
(
    [0] => Array
        (
            [0] => php programlama öğrenimi
            [1] => 2
        )
    [1] => Array
        (
            [0] => php pratikleri
            [1] => 2
        )
    [3] => Array
        (
            [0] => Eksersiz yapma
            [1] => 1
        )
    [4] => Array
        (
            [0] => PHP de önemli bir şeyler yazın
            [1] => 2
        )
)
```

- Çok boyutlu bir diziden öğe kaldırmak için unset() işlevi kullanabilirsiniz.

- Aşağıdaki örnek, \$gorevler dizisinin üçüncü öğesini kaldırmak için unset() işlevi kullanır:
- unset(\$gorevler[2]);
- 2 nolu yani "extra çalışma" öğesini kaldırmış olduk. Silme işleminden sonra dizi indexlerinin yeniden düzenlenmediğine dikkat edin!

PHP çok boyutlu diziden öge kaldırma (**array_splice**)

Dizinin **Önceki** hali

```
Array
(
    [0] => Array
        (
            [0] => php programlama öğrenimi
            [1] => 2
        )
    [1] => Array
        (
            [0] => php pratikleri
            [1] => 2
        )
    [2] => Array
        (
            [0] => extra çalışma
            [1] => 8
        )
    [3] => Array
        (
            [0] => Eksersiz yapma
            [1] => 1
        )
    [4] => Array
        (
            [0] => PHP de önemli bir şeyler yazın
            [1] => 2
        )
)
```

Dizinin **Sonraki** hali

```
Array
(
    [0] => Array
        (
            [0] => php programlama öğrenimi
            [1] => 2
        )
    [1] => Array
        (
            [0] => php pratikleri
            [1] => 2
        )
    [2] => Array
        (
            [0] => Eksersiz yapma
            [1] => 1
        )
    [3] => Array
        (
            [0] => PHP de önemli bir şeyler yazın
            [1] => 2
        )
)
```

- Çok boyutlu bir diziden öge kaldırmak için `array_splice` işlevi kullanabilirsiniz. Bu işlev silme işleminden sonra dizi indexlerini de düzenler.

- Yandaki örnek, `$gorevler` dizisinin üçüncü ögesini kaldırmak için `array_splice()` işlevi kullanır:
- `array_splice($gorevler,2,1);`
- 2. öğeden itibaren 1 öge yani "extra çalışma" ögesini kaldırmış olduk. Silme işleminden sonra dizi indexlerinin yeniden **düzenlendiğine** dikkat edin!

Foreach kullanarak çok boyutlu bir dizinin öğelerini listeleme

- Çok boyutlu bir diziyi yinelemek için, foreach deyimini iç içe geçmiş bir şekilde kullanırsınız:

- `<?php`
- `$gorevler = [`
- `['php programlama öğrenimi', 2],`
- `['php pratikleri', 2],`
- `['extra Çalışma', 8],`
- `['Eksersiz yapma', 1],`
- `];`
- `$gorevler[] = ['PHP de önemli bir şeyler yazın', 2];`

```
foreach ($gorevler as $gorev) {  
    • foreach ($gorev as $gorev_detay) {  
    •     echo $gorev_detay . ' saat ' . '<br>';  
    • }  
    • }  
    • ?>
```

```
php programlama öğrenimi saat  
2 saat  
php pratikleri saat  
2 saat  
extra Çalışma saat  
8 saat  
Eksersiz yapma saat  
1 saat  
PHP de önemli bir şeyler yazın saat  
2 saat
```

Not : ilk foreach çok boyutlu dizinin keylerini döndürür, 2nci foreach ise valueları döndürür.

Bazı Yararlı Dizi İşlevleri

- Bu bölümde, dizi işlemlerinde sıklıkla kullandığımız bir avuç dolusu yararlı dizi işlevleri göreceğiz.
- Count
- is_array
- in_array
- explode
- implode
- array_push
- array_pop

Count

- Bir dizideki eleman sayısını saymak için count işlevi kullanılır. Eğer bir diziyi for döngüsü ile yinelemek istiyorsanız bu oldukça kullanışlıdır.

```
<?php
$array = ['One', 'Two', 'Three'];

// dizinin eleman sayısını bulur
echo count($array);

// çıktı:
3

?>
```

is_array

- Diziler ile başa çıkmak için en yararlı işlevlerinden biridir. Bir değişkenin bir dizi veya başka bir veri türü olup olmadığını denetlemek için kullanılır.

```
$array = ['One', 'Two', 'Three'];

// dizimi?
if (is_array($array))
{
    // dizi ise yapılacak işlemler
}

?>
```

in_array

- Bir öğenin dizi içinde olup olmadığını denetlemek isterseniz, imdadınıza yetişecek olan in_array işlevidir.

```
<?php
$array = ['One', 'Two', 'Three'];

// dizide "one" varmı?
if (in_array('One', $array))
{
    echo 'Yes';
}
else
{
    echo 'No';
}
?>
```

in_array işlevinin ilk bağımsız değişkeni aradığınız öğe ve ikinci bağımsız değişken dizinin kendisidir.

array_key_exists

- Dizi içerisinde belirtilen anahtarın olup olmadığını kontrol eder.
- Yapısı (Syntax)
- `array_key_exists (mixed $key , array $array) : bool`
- **\$key** mixed
- Kontrol edilecek anahtarın adı
- **\$array** array
- Kontrol edilecek dizi
- **Sonuç** bool
- Dönen değer true yada false
- `$sonuç = array_key_exists('Kaydet', $_POST)`

```
• <?php
•     if(array_key_exists('button1', $_POST)) {
•         echo "1.butona bastınız";
•     }
•     else if(array_key_exists('button2',
$_POST)) {
•         echo "2.butona bastınız";
•     }
• }
• ?>
• <form method="post">
•     <input type="submit" name="button1"
•         class="button" value="Button1" />
•     <input type="submit" name="button2"
•         class="button" value="Button2" />
• </form>
•
```


Döngüler(program içinde tekrar eden işlemlerde kullanılır.)

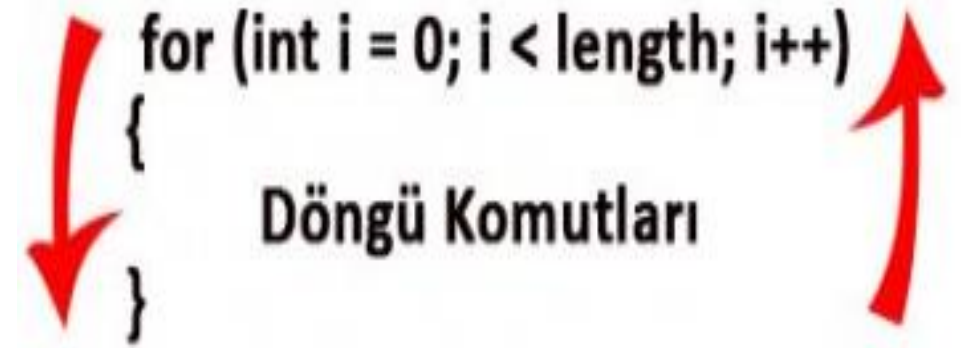
- Bazı durumlarda aynı kodu birçok kez çalıştırmak zorunda kalabiliriz.
- Bunun yanında birden fazla yapılacak işlem olduğunda aynı kod bloğunu sürekli tekrar yazmak programcılarının çok tercih ettiği bir yöntem değildir.
- Tekrar edilen kodlar yazdığınız programı ileride modifiye etmenizi zorlaştırırken, karışık bir görüntü oluşturarak kodların okunurluğunu azaltır.

- Bu bölümde;
 - For döngüsü
 1. for (\$i = 0; \$i < length; \$i++)
 2. {
 - 3.
 4. }
 - For each döngüsü
 1. foreach (dizi as eleman)
 2. {
 - 3.
 4. }
 - While döngüsü
 1. while (true)
 2. {
 - 3.
 4. }
 - Do While döngüsü
 1. do
 2. {
 - 3.
 4. }
 5. while (true)
- Konuları işlenecek.

For döngüsü

- Döngüler adından da anlaşılacağı gibi, sonlandırma koşulu sağlanıncaya kadar aynı komutları defalarca çalıştıran yapılardır.
- Php dilinde for döngüsü çok esnek bir yapıya sahip olup, neredeyse içerisinde döngü gerektiren tüm problemler bu komutla çözülebilir.
- *For ifadesinin genel formu aşağıdaki gibidir.*
- For(başlangıç değeri; koşul; değişim-miktarı)
- {
 - //Komutlar
- }

FOR DÖNGÜSÜ



```
for (int i = 0; i < length; i++)  
{  
    Döngü Komutları  
}
```

The diagram illustrates the execution flow of a for loop. A red arrow on the left points downwards from the opening curly brace to the closing curly brace, representing the body of the loop. Another red arrow on the right points upwards from the closing curly brace back to the increment part of the loop header, representing the update step.

for döngüsünün çalışma mantığını inceleyelim.

Başlangıç değeri; Bu kısım programın başlangıcında bir defaya mahsus olmak üzere çalıştırılır. Burada döngüyü kontrol eden değişkene bir başlangıç değeri atanır.

Koşul; Bu bölümde başlangıç değeri atanan değişken bir koşul ile test edilir eğer bu ifade *true* sonuç verirse, döngü bloğu içerisindeki komutlar çalıştırılır. Koşul *false* sonuç verirse döngü sonlandırılır.

Değişim-miktarı; Bu kısımda döngü değişkeni eksiltir veya arttırılır. Döngü her tekrarlandığında koşul değeri kontrol edilir, döngü bloğu çalıştırılır, döngü değişkeni arttırılır veya azaltır. Bu çalışma sistemi ta ki kontrol değişkeni koşulu sağlamaz hale gelene kadar devam eder.

Örnek: 1'den 10'a kadar olan sayıları alt alta ekrana yazdıralım.

- `for ($i = 1; $i <=10; $i++)`
- `{`
- `echo $i;`
- `}`

Örneğimizin çalışmasını adım adım inceleyelim.

Adım1: (`int i=1`) Kontrol değişkenimizin başlangıç değeri 1 olarak verilmiş.

Adım2: (`i<=10`) `i` değişkenimizin değeri koşulumuzla test ediliyor. “`i`” değeri 10 ve 10’dan küçük olduğu sürece döngü çalışacak.

Adım3: Koşul sağlandığı için döngü bloğuna girildi. Ekrana “`i`” değeri yani 1 yazdırıldı.

Adım4: (`i++`) Döngü bir kez çalıştıktan sonra `i++` ile `i` değeri 1 arttırıldı. Yani 2 oldu.

Adım5: Tekrar koşul kontrol edildi, “`i`” değeri 10’dan küçük olduğu için tekrar döngü bloğuna girildi.

Adım6: Bu işlem “`i`” değeri birer birer artıp 11 olana kadar devam eder. 11 olduğunda döngü bloğunun dışına çıkılır.

Örnek: 10'den 1'a kadar olan sayıları alt alta ekrana yazdıralım.

- `for ($i = 10; $i >=1; $i--)`
- `{`
- `echo $i;`
- `}`

Örneğimizin çalışmasını adım adım inceleyelim.

Adım1: (i=10) Kontrol değişkenimizin başlangıç değeri 10 olarak verilmiş.

Adım2: (i>=0) i değişkenimizin değeri koşulumuzla test ediliyor. “i” değeri 0 ve 0’dan büyük olduğu sürece döngü çalışacak.

Adım3: Koşul sağlandığı için döngü bloğuna girildi. Ekrana “i” değeri yani 10 yazdırıldı.

Adım4: (i--) Döngü bir kez çalıştıktan sonra i-- ile i değeri 1 azaltıldı. Yani 9 oldu.

Adım5: Tekrar koşul kontrol edildi, “i” değeri 0’dan küçük olduğu için tekrar döngü bloğuna girildi.

Adım6: Bu işlem “i” değeri birer birer azalıp 0 olana kadar devam eder.

a'dan z'ye kadar olan harfleri ekrana yazdıran program

1. `for($i = ord('a'); $i < ord('z'); $i++)`
2. `echo "char = ".chr($i)."
";`

Yada

```
<?php
for($i = ord('a');$i<ord('z');$i++)
echo "char = ".chr($i)."<br>";
echo "<hr>";
for($i = 'a';$i<'z';$i++)
echo "harf = ".$i." ----- ascii karşılığı = ".ord($i)."<br>";
```

ord fonksiyonu parametre olarak verilen karakterin ASCII değerini döndürür. chr()'nin tersini yapar.
chr() Fonksiyonu klavye üzerindeki her karakterin var olan ASCII karşılığını verir.

Z'den a'ya kadar olan harfleri ekrana yazdıran program

1. `for($i = ord('z'); $i < ord('a'); $i++)`
2. `echo "char = ".chr($i)."
";`

Tek ve çift sayıları tablo yapma

```
<?php
echo "<table border =1>";
echo "<tr><td>Tek Sayılar</td><td>Çift Sayılar</td></tr>";
for ($i=1;$i<100;$i++)
{
    $sayi=rand(100,450); // 100 ile 450 arasında rastgele sayı üretir.
    echo "<tr>";
    echo "<td>".(($sayi % 2 ==1)?$sayi:"")."</td>"; echo "<td>".(($sayi % 2 ==0)?$sayi:"")."</td>";
    echo "</tr>";
}
```

Tek ve çift sayıları düzenli tablo yapma

```
echo "<table border =1>";
echo "<tr><td>Tek Sayılar</td><td>Çift Sayılar</td></tr>";
$tek=array();
$cift=array();
for ($i=1;$i<100;$i++) {
    $sayi = rand(100, 450);
    if($sayi % 2==0)
        $cift[]=$sayi ;
    else
        $tek[]=$sayi;
}
// iki diziden eleman sayısı en büyük olanı buldurmuş oluyoruz..
if(count($tek) > count($cift))
    $ebs=count($tek);
else
    $ebs=count($cift);
for ($i=1;$i<$ebs;$i++) {
    echo "<tr>";
    echo "<td>".$tek[$i]."</td><td>".$cift[$i]."</td></tr>";
}
```

For döngüsü ile FAKTÖRİYEL hesaplama

- <?php
- \$sayi =5;
- \$sonuc = 1;
- for (\$i = 1; \$i <= \$sayi; \$i++)
- {
- echo "\$sonuc * \$i = " . (\$sonuc * \$i). "
";
- \$sonuc = \$sonuc * \$i;
- }
- ?>

- Bir sayının faktöriyeli; 1'den sayının kendisine kadar olan sayıların çarpımıdır.

Bunun için ihtiyacımız olan faktöriyel hesaplanması istenilen sayıyı tutan bir değişken ve faktöriyel hesabını tutacak başka bir değişkendir.

- Daha sonra 1'den sayının kendisine kadar bir for döngüsü oluşturacağız ve faktöriyel hesabını gerçekleştireceğiz.

Not: sonuc değişkeninin ilk değeri 1 olarak atanmıştır. Eğer ilk değeri 1 olmazsa (yani ilk değer verilmez veya 0 olarak sayaçmış gibi düşünülürse) faktöriyel işleminin sonucu her seferinde 0 çıkacaktır.

Tek ve çift sayıları yazdıran prg.

Tek sayıları ekrana yazdıran prg

```
1. for (int $i = 1; $i <= 10; $i += 2)
2.     {
3.     Echo $i ;
4.     }
```

Çift sayıları ekrana yazdıran prg

```
1. for ($i = 2; $i <= 10; $i += 2)
2.     {
3.     echo "sayı : ", $i ;
4.     }
```

Eğer «if else» karar yapılarını biliyorsak yukarıdaki 2 örneğimizi yandaki gibi de yazabilirdik;

```
• for ($i = 0; $ i <= 10; $ i++)
•     {
•         if ($i % 2==0) //sayının 2'ye bölümünde kalan 0 ise çifttir
•             echo "sayı çifttir : ", $i;
•         else
•             echo "sayı tekdir : ", $i;
•     }
```

Döngülerde BREAK kullanımı

For döngüsünden herhangi bir anda çıkış için **break** komutunu kullanabiliriz.

Bir döngü yapısı içinden, döngüyü kontrol eden koşul ifadesini beklemeksizin döngü dışına çıkmak için kullanılır.

break deyimi bir döngü içinde yer almışsa bu durumda *break* deyimi ile karşılaşır karşılaşmaz döngü dışına çıkılır ve döngüyü izleyen deyime geçilir.

```
1. //20 tane rastgele oluşturulan sayıların
2. //toplamını bulan prg.Eğer 0'dan küçük bir sayı gelirse
   //döngü bitsin
3. $sayi=0;
4. $toplam=0;
5. for($i=1;$i<20;$i++)
6. {
7.     $sayi=rand(-10,100);
8.     if ($sayi<0)
9.     {
10.         echo "Döngüden çıkılıyor...";
11.         break;
12.     }
13.
14.     else
15.         $toplam+=$sayi;
16. }
17. echo "Toplam : $toplam","<br>Döngü $i defa döndü";
```

Continue kullanımı

Bir döngü içerisinde continue deyimi ile karşılaşılsa, ondan sonra gelen deyimler atlanır ve döngü bir sonraki çevrime girer.

Örnek: rastgele oluşturulan 100 sayıdan tek olanları toplayan çift olanlarda ise döngüyü *continue* deyimiyle sonraki çevrimine yönlendiren kodları yazalım.

```
• <?php
• $toplam=0;
• for ($i=1;$i<100;$i++)
• {
•     $sayi=rand(0,500);
•     if($sayi%2==0)
•     {
•         echo "sayı çift olduğu için toplama <span style=color:red> yapılmadı : $sayi</span></br>";
•         continue;
•     }
•     else
•     {
•         $toplam+=$sayi;
•         echo "sayı : $sayi, toplam : $toplam</br>";
•     }
• }
• ?>
```

Alttaki program kodları sayfaya 5 satır ve 3 sütundan oluşan bir tablo çizer.

- echo "<table border='1' width='200px'>";
- for (\$i=1;\$i<=5;\$i++){
- echo "<tr >";
- echo "<td> </td>";
- echo "<td> </td>";
- echo "<td> </td>";
- echo "</tr>";
- }
- echo "</table>";
- echo "<table border='1' width='200px'>";
- for (\$i=1;\$i<=10;\$i++){
- echo "<tr
bgcolor='',(\$i%2)?'#abda68':'#d0f896','>";
- echo "<td> </td>";
- echo "<td> </td>";
- echo "</tr>";
- }
- echo "</table>";

Dizi elemanlarını for döngüsü ile yazdırma

Adı	Soyadı	Numara	Vize	Final
Şenol	Demirci	109020	95	90
Şahin	Sağlam	109021	85	90
Ayhan	Pirinç	109022	80	70
Kasım	Gürpınar	109023	70	75

Not: Final sütunundan sonra ORTALAMA'yı hesaplatıp yazdırmayı deneyin.

- `<?php`
- `$dizi=array();`
- `$dizi[]=array('ad'=>'Şenol','soyad'=>'Demirci','ogrno'=>109020,'vize'=>95,'final'=>90);`
- `$dizi[]=array('ad'=>'Şahin','soyad'=>'Sağlam','ogrno'=>109021,'vize'=>85,'final'=>90);`
- `$dizi[]=array('ad'=>'Ayhan','soyad'=>'Pirinç','ogrno'=>109022,'vize'=>80,'final'=>70);`
- `$dizi[]=array('ad'=>'Kasım','soyad'=>'Gürpınar','ogrno'=>109023,'vize'=>70,'final'=>75);`
- `echo "<table border=1>";`
- `echo`
`"<tr><th>Adı</th><th>Soyadı</th><th>Numara</th><th>Vize</th><th>Final</th></tr>";`
- `$renk="white";`
- `for($i=0;$i<count($dizi);$i++)`
- `{`
- `($i%2==0)?$renk="gray":$renk="red";`
- `echo "<tr style=background-color: ".$renk."> <td>".$dizi[$i]['ad']. "</td>".$dizi[$i]['soyad']. "</td>".`
- `"<td>".$dizi[$i]['ogrno']. "</td><td>".$dizi[$i]['vize']. "</td><td>".$dizi[$i]['final']. "</td></tr>";`
- `}echo "</table>"`
- `?>`

While döngüsü

- While döngüsü, belirlenen komut bloğunu koşul sağlandığı sürece çalıştıran temel döngü yapılarından bir tanesidir. While döngüsünün genel formu aşağıdaki gibidir.
- While(koşul)
- {
- //Döngü gövdesi-Komutlar
- }
- Döngü gövdesi, koşul deyimi true olduğu sürece çalıştırılır, koşul false olduğunda ise döngüden çıkılır. While döngüsünde döngü değişkeni bulunmaz, bunun yerine while bloğuna girmeden başlangıçta değeri bilinen bir değişken kullanılır. Koşul ifadesiyle ilişkili olan bu değişkenin değeri döngü bloğu içerisinde değiştirilir.
- while döngüsünü başlangıç ve bitiş değeri belli olmayan durumlarda kullanılması daha uygundur. Bir işlemin kaç defa yapılacağı belli olmayan durumlara başlangıç ve bitiş belli olmayan durumlar diyebiliriz.

İlk örneğimizde ekrana 1'den 10'a kadar olan sayıları alt alta yazdıralım.(while döngüsü)

- <?php
- \$sayi=1;
- while(\$sayi<=10){
- echo "Sayı \$sayi
";
- \$sayi++;
- }
- ?>
- **while** için söylenen koşul **\$sayi** değişkeninin 10'a eşit yada 10'dan küçük olma koşuludur.
- Dolayısıyla döngüye girebilmek için bu koşula uygun bir değeri başlangıçta \$sayi=1; diyerek tanımlıyoruz.
- Daha sonra yapmasını istediğimiz kodu yazıp **\$sayi** değişkenini 1 arttırdık. Döngü koşulu **\$sayi** değişkenine bağlı çalışmaktadır.
- Eğer **\$sayi** değişkenini 1 arttırmazsak değeri hep 1 olarak kalır ki bu döngünün sonsuz döngüye yani kısır döngüye girmesine sebep olur. Döngü her tekrarında koşulu kontrol eder. Koşul sağlanmadığı anda döngüden çıkar.

1 ile 100 arasında rastgele üretilen sayı bir önceki sayıya eşit olduğunda duran php kodlarını yazalım.(while döngüsü)

- <?php
- \$onceki=rand(1,100);
- echo \$onceki," ";
- while(true){ // while, koşul true olduğu sürece döner
 - \$yeni=rand(1,100);
 - echo \$yeni," ";
 - if (\$onceki==\$yeni)
 - break;
 - \$onceki=\$yeni;
- }
- ?>
- while, yapısı itibarıyla koşul sağlandığı sürece yani true olduğu sürece döner.
- while(true) diyerek döngünün devamlı dönmesi sağlanıyor.
- Yani sonsuz dön anlamına gelir.
- Bu tür döngülerden çıkmak için **break** komutu kullanılır.

1 ile 100 arasında rastgele üretilen 10 tek sayıyı bir diziye atayıp ekrana yazdıralım.

- <?php
- \$sayac=1;
- while(\$sayac<=10){
- \$sayi=rand(1,100);
- if (\$sayi%2==1){ // üretilen sayı tek ise
- \$sayilar[]=\$sayi;
- \$sayac++;
- }
- }
- \$indis=0;
- while(@\$sayilar[\$indis]){
- echo \$sayilar[\$indis]," ";
- \$indis++;
- }
- ?>

Do while döngüsü

- while do döngüsünde şart baştır ve şart doğru değilse döngü hiç başlatılmayacaktır. Döngünün bir kez işletilip, sonra şartın sorgulanmasının gerektiği durumlarda do while döngülerini kullanırız. Döngünün kullanımı şu şekildedir :

```
do {  
işletilecek kodlar;  
} while ( şart );
```

While döngüsünde önce şart ifadesine bakılır sağlanırsa döngüye girer ve tur atmaya başlar. Fakat do-while döngüsünde başlangıçta şart koşulmadığı için döngü her şartta 1 tur atar. Arada farkı anlamak için küçük bir örnek yapalım.

- <?php
- \$i=5;
- while(\$i>10){
- echo "Döngü içerisine girildi.";
- }
- ?>
- //Bu kod çalıştırıldığında tarayıcıda hiç birşey görünmez.

- <?php
- \$i=5;
- do{
- echo "Döngü içerisine girildi.";
- } while(\$i>10);
- ?>
- //Bu kodun çıktısı olarak tarayıcıda "Döngü içerisine girildi." ifadesini görürsünüz. Yani ne olursa olsun döngü 1 defa çalışır.

1 ile 100 arasında rastgele 50 olana kadar sayı üretelim ve kaç defada 50 sayısını bulduğunu ekrana yazalım.

- <?php
- \$i=0;
- do{
- \$sayi=rand(1,100);
- \$i++;
- }while(\$sayi!=50);
- echo "Kaç tahminde buldun: \$i";
- ?>